



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

DEPARTMENT OF INFORMATION SYSTEMS

MONITOROVÁNÍ BĚHU OS LINUX

LINUX OS MONITORING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

JURAJ KORČEK

VEDOUcí PRÁCE

SUPERVISOR

Mgr. ROMAN TRCHALÍK, Ph.D.

BRNO 2018

Zadání bakalářské práce

Řešitel: **Korček Juraj**

Obor: Informační technologie

Téma: **Monitorování běhu OS Linux
Linux OS Monitoring**

Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s nástroji na monitorování chodu OS, zejména technologií zmíněných v bodě 3.
2. Navrhněte nástroj nebo sadu skriptů na monitorování serverových clusterů. Zaměřte na logy základních služeb OS, uživatelských aplikací a monitorování HW. Při abnormálním chování aplikace, systému nebo chybě HW proveďte specifickou akci (notifikaci, pročištění složky, zablokování účtu, apod.)
3. Navržené řešení implementujte. Do implementace zahrňte i nasezení monitorovacího nástroje pomocí ANSIBLE nebo vlastního skriptu s knihovnou Python Fabric.
4. Proveďte testování Vašeho řešení a jeho vyhodnocení.

Literatura:

- Ansible Documentation. Red Hat, Inc. [online]. Rev. 2017-10-01 [cit. 2017-10-01]. Available at URL: < <http://docs.ansible.com/>>.
- Fabric - Pythonic Remote Execution. Jeff Forcier. 2017 [cit. 2017-10-01]. Available at URL: < <http://docs.fabfile.org/>>.
- Dále podle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese

<http://www.fit.vutbr.cz/info/szz/>

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Trchalík Roman, Mgr., Ph.D., UIFS FIT VUT**

Datum zadání: 1. listopadu 2017

Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
602 00 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cieľom tejto bakalárskej práce je návrh a následná implementácia monitorovacieho nástroja pre operačné systémy GNU/Linux. Tento nástroj sleduje jednotlivé logy a taktiež vyťaženia zdrojov a pri zistení chyby, vysokého zaťaženia alebo abnormálneho správania systému notifikuje systémového administrátora o všetkých dôležitých incidentoch. Výsledný produkt je určený hlavne pre systémových administrátorov, ktorým zjednoduší prácu tým, že ich upozorňuje iba na dôležité zmeny v systéme. Nástroj je implementovaný v jazyku Python a je rozdelený do menších nezávislých skriptov z dôvodu ľahšej implementácie nových skriptov. Tento programovací jazyk bol zvolený kvôli bezproblémovej podpore medzi rôznymi distribúciami GNU/Linux.

Abstract

The aim of this bachelor's thesis is design and implementation of monitoring tool for GNU/Linux operating systems. This program monitors logs created by system, system load and computer resources. It notifies system administrator in case of high system load, abnormal behavior or when an error occurs. The resulting product is especially aimed for system administrators of GNU/Linux, whose work will be simplified due to this software utility, which sends notification of only important system changes. It is implemented in scripting language Python and divided into smaller independent scripts for easy implementation of new scripts in the future. This programming language was chosen to ensure operation among all GNU/Linux distributions.

Klíčové slová

unix, linux, monitorovanie, syslog, server, cluster, python

Keywords

unix, linux, monitoring, syslog, server, cluster, python

Citácia

KORČEK, Juraj. *Monitorování běhu OS Linux*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Roman Trchalík, Ph.D.

Monitorování běhu OS Linux

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Mgr. Romana Trchalíka, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....

Juraj Korček

10. mája 2018

Podakovanie

Týmto by som sa chcel poďakovať vedúcemu práce Mgr. Romanovi Trchalíkovi, Ph.D. za odborné vedenie, cenné rady a poskytnutý čas, ktorý mi pri tvorbe práce venoval.

Obsah

1	Úvod	3
2	Teória	4
2.1	Operačný systém	4
2.2	Jadro OS	4
2.3	Unix	4
2.4	GNU/Linux	5
2.5	Distribúcie	5
2.6	Cluster	6
2.7	Python	9
2.8	Nasadzovací softvér	9
2.9	Ansible	10
2.10	Python Fabric	12
3	Monitorovanie systému	14
3.1	Textové utility na monitorovanie	14
3.2	Syslog	17
3.3	Nagios	18
3.4	Zabbix	19
4	Návrh	20
4.1	Požiadavky na aplikáciu	20
4.2	Odlišnosti oproti existujúcim riešeniam	22
4.3	Inštalácia	22
4.4	Hlavný skript	24
5	Implementácia	27
5.1	Hlavný skript Linmon	27
5.2	Moduly monitorujúce zdroje a služby	34
5.3	Moduly monitorujúce rsyslog záznamy	35
5.4	Podporné skripty	36
5.5	Nasadenie pomocou Fabric3	36
6	Testovanie	37
6.1	Testovacie prostredie	37
6.2	Testovanie mechanizmu vysokej dostupnosti	40
6.3	Filtrovanie notifikačných správ	40
6.4	Testovanie módu údržby	42

7 Záver	43
Literatúra	44
A Zoznam a činnosť implementovaných skriptov	46
B Obsah CD	49

Kapitola 1

Úvod

Linux je slobodný open source operačný systém obľúbený vďaka výbornej podpore ľudí z komunity, možnosti modifikovania zdrojových kódov a v neposlednom rade širokého spektra dostupného bezplatného softvéru. Vďaka týmto, ale aj mnohým iným vlastnostiam a výhodám ho môžeme nájsť na väčšine superpočítačoch a serveroch.

Častokrát sú tieto servery spájané do väčších celkov a vytvárajú serverové clustery a tie zasa serverové farmy. K takejto agregácii dochádza z viacerých dôvodov, napríklad pre zaisťenie dostupnosti služby, zvýšenie výpočtového výkonu a podobne. Takýto cluster môže obsahovať desiatky až stovky serverov, pričom treba zaistiť ich bezproblémový chod.

Každé z týchto zariadení obsahuje určité požiadavky, a teda môže byť rôzne zaťažené prípadne sa na ňom môže vyskytnúť porucha alebo môže byť taktiež cieľom útočníkov. Preto je nevyhnutné mať pod dohľadom každý uzol, monitorovať jednotlivé operácie, ktoré spracováva pre optimálne rozloženie zaťaženia medzi jednotlivé uzly clusteru, predvídať poddimenzovanosť uzlov z hľadiska výkonu a detekovať možné problémy zavčas, ešte pred koncovým užívateľom. V závislosti na počte serverov sú používané rôzne techniky monitorovania. V prípade malých počtov serverov systémový administrátor využíva vlastné vzájomne nezávislé skripty, no v prípade serverových fariem sa používajú zväčša monitorovacie nástroje ako Nagios, Zabbix a im podobné programy.

Táto bakalárska práca vznikla z dôvodu skombinovania výhod oboch prístupov monitorovania systému s ohľadom na využitie už existujúcich skriptov s ich minimálnou modifikáciou. Výsledný monitorovací program bude rozdelený do menších skriptov, ktoré budú môcť byť použité aj samostatne. Tento prístup je zvolený z dôvodu modularity a flexibility výsledného nástroja.

V nasledujúcej kapitole 2 je opísaná motivácia použitia operačných systémov, v skratke je opísaný systém GNU/Linux a teória k počítačovému clusteru. V ďalšej kapitole 3 pojednávajúcej o monitorovaní sa bližšie pozrieme na používané monitorovacie nástroje Nagios a Zabbix a protokol syslog. V kapitole 4 si popíšeme návrh monitorovacieho nástroja s opisom jeho kľúčových vlastností, hierarchického rozdelenia skriptov a rozdielom oproti existujúcim riešeniam. Kapitole 5 sa pozrieme na zaujímavé prvky skriptov, formát notificačnej správy a taktiež na implementáciu nasadzovacieho skriptu. Posledná kapitola 6 opisuje testovanie, a to predovšetkým testovacie prostredia a testovanie kľúčových prvkov programu.

Kapitola 2

Teória

2.1 Operačný systém

Operačný systém je základom pre každý výpočtový systém. Je to počítačový program, ktorý vytvára spojujúcu medzivrstvu medzi hardvérom, užívateľmi a ich užívateľským softvérom. Medzi hlavné role operačných systémov patrí správa prostriedkov a taktiež zabezpečenie prostredia užívateľom pre bezproblémový prístup k užívateľským programom. Operačný systém by mal spravovať prostriedky s ohľadom na efektivitu a bezpečnosť. Prostriedky umožňuje spravodlivo rozdeľovať medzi užívateľov, užívateľský softvér a teda dáva možnosť viacerým procesom zdieľať rovnaký procesor, priestor vo volatilných a nevolatilných pamätiach. Druhá zmienená rola je veľmi dôležitá a k jej zaisteniu je nutné jednotné rozhranie, ktoré zabezpečuje prenositeľnosť jednotlivých užívateľských aplikácií. Operačný systém poskytuje používateľom abstrakciu a teda pre užívateľa zakrýva nízkoúrovňové operácie. [14]

Presná definícia, čo všetko je súčasťou operačného systému nie je jednotná, niekedy sa tým myslí iba jadro, inokedy okrem jadra tiež grafické alebo textové rozhranie, systémové aplikačné programy a ovládače.

2.2 Jadro OS

Jadro operačného systému nazývané tiež kernel je najnižšia a najzákladnejšia časť operačného systému. Kernel sa zavádza ako prvý a beží počas celej doby chodu operačného systému, pričom je uložený v operačnej pamäti a podľa potrieb spúšťa ostatné súčasti operačného systému. Pre užívateľa a užívateľský softvér kompletne zapuzdruje hardvér tým, že na neho priamo naväzuje. Zariaďuje základnú správu prostriedkov a to prevažne správu kontextu a nastavenie hardvéru. Tým, že obvykle beží v privilegovanom režime, tak má neobmedzený prístup k hardvéru a môže nad ním vykonávať rôzne operácie. Počet služieb, ktoré jadro poskytuje zväčša závisí na kompromise medzi viacerými faktormi a to hlavne bezpečnosťou, efektivitou a flexibilitou. [14]

2.3 Unix

Unix je univerzálny, viac užívateľský, viacúlohový operačný systém vyvinutý v 70. – tých rokoch minulého storočia [21]. Tento operačný systém bol vyvinutý v spoločnosti Bell Labs Kenom Thompsonom a Dennisom Ritchiem [21]. Ide o prvý prenositeľný operačný systém, pričom tejto vlastnosti vďačí programovaciemu jazyku C, do ktorého bol po čase prepísaný

a ktorého tvorcom je Dennis Ritchie. Vďaka tomu ho bolo možné takmer od počiatku použiť na širokej škále zariadení resp. platforiem a predurčilo ho k ďalšiemu úspechu. Unix je veľmi obľúbený v akademickej sfére hlavne z dôvodu otvoreného kódu, bezplatnosti a spoľahlivosti.

Výhody Unix-u:

- flexibilita – môže byť inštalovaný od vstavaných zariadení až po superpočítače.
- stabilita – oproti mnohým iným operačným systémom je prevádzkyschopný s dostatočným výkonom napriek absencii reštartu v porovnaní s MS Windows.
- bezpečnosť – pokročilé nastavenie prístupových práv a SELinux.
- bezplatnosť – minimalizuje počiatočné náklady za licencie.
- open-source – každý môže nahliadnuť do zdrojových kódov a modifikovať ich.

Od vzniku Unix-u bolo uvedených mnoho derivátov, ako napríklad [9]:

- BSD
- Solaris
- GNU/Linux
- macOS

2.4 GNU/Linux

GNU je operačný systém inšpirovaný UNIX-om patriaci do skupiny UNIX-like OS, ktorého vývoj započal v roku 1984 Richardom Stallmanom na MIT. Myšlienka a filozofia tohto operačného systému je sloboda užívateľov spúšťať, kopírovať, distribuovať, študovať a vylepšovať softvér. Motiváciou bolo vytvoriť operačný systém z čisto slobodného softvéru. Operačný systém GNU využíva jadro Linux, ktoré bolo spočiatku vyvíjané fínskym študentom Linusom Torvaldsom a nahradilo jadro Hurd vyvíjané projektom GNU. [15]

V súčasnosti je GNU/Linux vyvíjaný za pomoci komunity vývojárov, v ktorej sú jednotlivci, ako aj veľké korporácie z celého sveta, napríklad Red Hat, Intel, IBM, SUSE [15].

Operačný systém GNU/Linux je vyvíjaný s ohľadom na podporu väčšiny softvéru vyvinutého pre Unix. Medzi najznámejšie patria Bourne shell, systém X Window, mkfs, fsck a mnohé ďalšie.

Od svojho vzniku si GNU/Linux udržuje rastúce tempo v podiele zastúpenia operačných systémov. Bezkonkurenčne vedie v percentuálnom nasadení na superpočítačoch [11], kde ostatné operačné systémy sú v minoritnom zastúpení. Taktiež má výrazný náskok v oblasti webových serverov [18], no jeho podiel na osobných počítačoch alebo pracovných staniciach je okolo 5%.

2.5 Distribúcie

Linuxová distribúcia je operačný systém pozostávajúci z kolekcie softvérového vybavenia a správcu balíkov založený na Linuxovom jadre [2]. Pred inštaláciou samotného operačného

systému by sme si mali vybrať vhodnú distribúciu pre dané zariadenie z hľadiska nárokov na hardvér, počtu súčastí distribúcie, podpory zo strany komunity a prostredia nasadenia.

Distribúcie môžu byť ihneď pripravené k použitiu a to napríklad spustením z USB kľúča, nainštalovaním na pevný disk alebo sú distribuované ako zdrojový kód a je ich nutné skompilovať.

Primárne sa distribúcie rozdeľujú podľa prostredia nasadenia a to na distribúcie určené pre osobné počítače, pracovné stanice a tie, ktoré sú určené pre nasadenie na serveroch [2]. Ich hlavným rozdielom je množstvo zabraného miesta na disku, prítomnosť grafického užívateľského rozhrania a počtu obsiahnutých balíkov. Distribúcie určené pre serveri sú zväčša zamerané na výkon a teda typicky neobsahujú grafické užívateľské rozhranie a obsahujú iba to najnutnejšie, pričom všetky potrebné balíčky je možné dodatočne doinštalovať. Pokročilí užívatelia a administrátori si môžu vytvoriť vlastnú distribúciu, ktorá bude zahŕňať kľúčové balíčky a súčasti operačného systému pre konkrétne nasadenie, čím sa zvýši efektivita a zároveň bude hardvér menej zaťažný.

Populárne distribúcie pre PC/Workstation:

- Ubuntu
- Debian
- Fedora
- Arch
- Mint

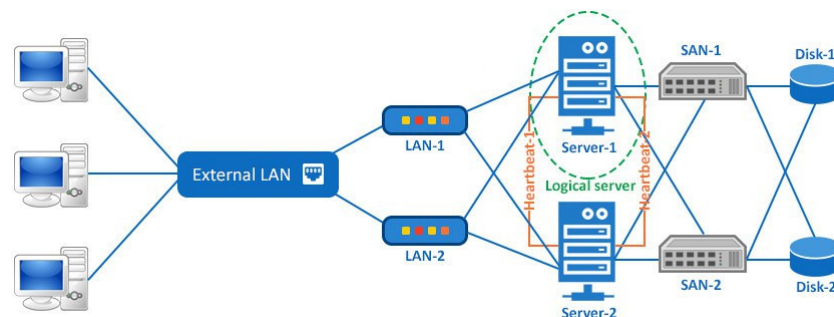
Populárne distribúcie pre serverové nasadenie:

- Red Hat Enterprise Linux
- Ubuntu Server
- Centos
- SUSE Enterprise Linux
- Debian

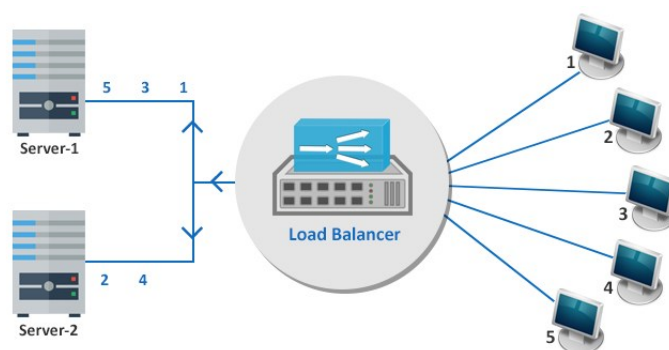
2.6 Cluster

Požiadavky na výkon počítačov neustále stúpajú, preto je nevyhnutnosťou namiesto jedného serveru, ktorý by náročné úlohy a záťaž nezvládal, zoskupiť viacero počítačov a tým urýchliť výpočtové operácie. Práve tento dôvod viedol k vytvoreniu počítačového clusteru.

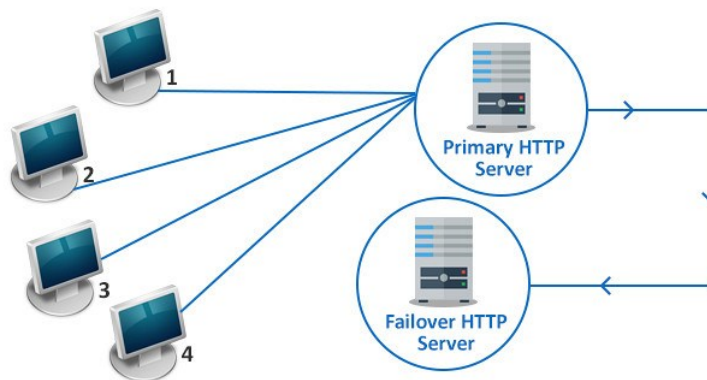
Počítačový cluster je zoskupenie aspoň dvoch počítačov (uzlov), ktoré spolu spolupracujú a navonok sa tvária ako jeden počítač [3]. Dôvody pre nasadenie takéhoto riešenia sú rôzne, pričom zväčša ide o navýšenie výpočtového výkonu, zvýšenie spoľahlivosti a prerozdelenie záťaže.



Obr. 2.1: Cluster s vysokou dostupnosťou [17]



Obr. 2.2: Active-Active Cluster [17]



Obr. 2.3: Active-Passive Cluster [17]

Typy clusterov

Z hľadiska miesta nasadenia a zverených úloh pre výpočet a role sú počítačové clustery rozdelené do viacerých kategórií. V závislosti na použitom operačnom systéme sú častými zástupcami Windows cluster a Linux cluster.

Typy clusterov [3]:

- Cluster s vysokou dostupnosťou
- Cluster s rozložením záťaže
- Výpočtový cluster
- Úložný cluster

Cluster s vysokou dostupnosťou

Po určitej dobe prevádzky hardvér počítača degraduje, preto je nutné počítať s výpadkom služieb bežiacich na serveri. Z tohto dôvodu vznikol cluster s vysokou dostupnosťou alebo inak nazývaný High availability cluster. V prípade výpadku služby na niektorom z uzlov, preberie iný uzol, ktorý poskytuje tú istú službu jeho rolu. Tým je zaistená dostupnosť služby. Tento koncept taktiež umožňuje vykonávať plánovaný servis bez toho, aby zákazník pocítil výpadok alebo zníženie výkonu. [3]

Cluster s rozložením záťaže

Tento typ clusteru využíva metódu, kedy sú dáta distribuované medzi dvoma a viac servermi, pričom dáta na všetkých serveroch by mali byť totožné. Počíta sa s ideálnym rozložením záťaže pre jednotlivé uzly. Typicky je tento typ clusterov často používaný pre web servery, kde zaisťuje vysokú rýchlosť a zároveň garanciu dostupnosti kvôli redundancii. V prípade dvoch serverov vytvárajúcich tento typ clusteru, je jeden určený za primárny a ďalší za sekundárny, pričom záťaž medzi tieto dva uzly rozdeľuje tzv. Master load balancer. Jednou z nevýhod tohto typu clusteru je nutnosť replikácie dát a teda redundancie, čo sťažuje prácu pre zaistenie konzistencie dát uložených na oboch uzloch a taktiež niekoľkonásobné finančné zaťaženie v závislosti na počte uzlov. [3]

Výpočtový cluster

Výpočtový cluster využíva sa na náročné výpočty, kde by jeden vysokovýkonný počítač bol priveľmi drahý. Takýto cluster sa skladá z aspoň dvoch počítačov pričom nemusia byť veľmi výkonné, no sú navzájom prepojené vysokorýchlostnou sieťou a zdieľajú medzi sebou výpočtové zdroje. [3]

Úložný cluster

Tento cluster patrí k špeciálnym typom clusteru s rozložením záťaže, ktorý zaisťuje prístup k diskovej kapacite rozdelenej medzi viacero počítačov kvôli vyššej prenosovej rýchlosti a zaisteniu spoľahlivosti. Používa špeciálny súborový systém, ktorého príkladom je GFS od spoločnosti Red Hat zaisťujúci redundanciu, konzistenciu, integritu dát, mechanizmus zamykania súborov a pokrytie výpadkov. [3]

Nevyhnutné prvky na vytvorenie clusteru

Počítačový cluster vzniká vtedy, ak sú vzájomne prepojené aspoň dva počítače, ideálne pre zaistenie dostupnosti a vyváženého výkonu je však odporúčané použiť viacej uzlov.

Druhým predpokladom je prepojenie počítačov vysokorýchlostnou počítačovou sieťou, ktorá zabezpečí vysoký tok dát a to najmä pri implementácii webového clusteru s rozložením záťaže. Posledným krokom je voľba správcu zdrojov clusteru, ktorý sa postará o prerozdeľovanie záťaže, nakonfigurovanie jednej spoločnej IP adresy pre uzly v clusteri a vykoná ďalšie nastavenia potrebné pre prevádzku. [3]

Správcovia zdrojov clusterov

Každý uzol clusteru musí byť spravovaný nejakým programom, ktorý zaistí migráciu služby na iný uzol v prípade poruchy alebo zvýšeného dopytu po službe, prípadne riadenie rozloženia záťaže. Tento softvér sa nazýva správca zdrojov clusteru. Príkladmi takéhoto softvéru sú Pacemaker, oneSIS, OpenHPC, Linux-HA.

2.7 Python

Programovací jazyk Python je interpretovaný, interaktívny, objektovo orientovaný programovací jazyk vytvorený holanďanom Guidom van Rossum [6]. Jeho veľkou výhodou je možnosť využitia na širokom spektre operačných systémov a taktiež fakt, že je vyvíjaný ako open source projekt.

Syntax Pythonu je jedným z dôvodov, prečo sa stal tak populárnym. Je totiž navrhovaný s ohľadom na čitateľnosť a syntax je veľmi podobná programovaciemu jazyku C. Veľkým rozdielom oproti iným jazykom je spôsob zápisu blokov kódu, ktorý je daný odsadením a nie ako v jazyku C pomocou zátvoriek alebo v Pascale kľúčovými slovami **Begin** a **End**.

V súčasnosti sa na tvorbu programov používajú dve verzie a to Python 2.x a Python 3.x. Aj keď by sa mohlo zdať, že ide o ten istý programovací jazyk, niektoré syntaktické konštrukcie nie sú kompatibilné medzi oboma verziami [6]. V neprospech Python 3.x hovorí aj to, že mnohé knižnice sú určené pre Python 2.x a teda často úplne nepoužiteľné. Nedávno zverejnené prehlásenie hovorí o konci vývoja a podpory Python 2.x, preto pre budúce programy je výhodnejšie použiť Python 3.x¹.

2.8 Nasadzovací softvér

Každý program, ktorý chceme používať je v prvom rade nutné nainštalovať a nakonfigurovať pre správne fungovanie na zariadení a pre konkrétne použitie. Pri nasadení, teda inštalácii a konfigurácii programu na jednom zariadení nie je problém tieto kroky aplikovať manuálne, problémom však je nasadenie programu alebo programov na desiatkach až stovkách zariadení. Pre tento účel vznikli špecializované programy, ktoré zjednodušujú nasadenie aplikácií vo výpočtových centrách s množstvami počítačov a clusterov.

Nasadenie softvéru je súbor procesov alebo aktivít, ktoré vedú k dostupnosti požadovaného inštalovaného softvéru [19]. Špecializované programy a knižnice programovacích jazykov umožňujú nielen samotné nasadenie, teda inštaláciu a konfiguráciu programu, ale aj ďalšie dôležité aktivity medzi ktoré patrí aktualizácia, odinštalácia a sledovanie nainštalovaných verzií [19]. Je treba si uvedomiť, že tento softvér okrem uľahčenia práce znižuje riziko vzniku rôznych chýb pri aktualizáciách a nasadení. Dôležitou vlastnosťou je taktiež stránka bezpečnosti, ktorú napomáha zlepšovať spomínaným sledovaním verzií a hromadnou aktualizáciou na všetkých staniciach. Namiesto manuálnych aktualizácií, ktoré pri desiatkach

¹<http://legacy.Python.org/dev/peps/pep-0373/>

až stovkách uzloch môžu nakopiť chyby a zároveň sú časovo náročné, aktualizácia pomocou tohto softvéru je vykonávaná hromadne na zvolených zariadeniach s rovnakými podmienkami a postupom.

Pridanou hodnotou nasadzovacieho softvéru je okrem ušetreného času taktiež zníženie nákladov a rýchlejšia dostupnosť nasadzovanej služby v kombinácii s nižším rizikom vnesenia chýb.

Nasadzovacieho softvéru existuje celá rada, pričom sa líšia ich primárnym určením. Nástroje Travis a Jenkins sú primárne určené na automatizované testovanie, no zvládnu aj nasadenie otestovaných programov. Medzi ďalšie nástroje, respektíve knižnice určené na nasadenie patria napríklad Python Fabric a Capistrano. Najuniverzálnejšími a veľmi mocnými nástrojmi na nasadenie, konfiguráciu, aktualizáciu sú Ansible od firmy Red Hat, ale aj Puppet alebo Salt.

2.9 Ansible

Ansible je program na automatizáciu nasadenia aplikácií a správu konfigurácií [12]. Jeho veľkou devízou je multiplatformnosť, je ho možné používať ako na staniciach s operačným systémom Linux, tak aj na Microsoft Windows. Konfiguračný súbor tzv. Ansible Playbook je veľmi dobre čitateľný pre používateľa aj stroj a nevyžaduje špeciálne programátorské nároky. Ansible je napísaný v programovacom jazyku Python pre operačné systémy Linux a v PowerShell pre operačný systém Windows [12].

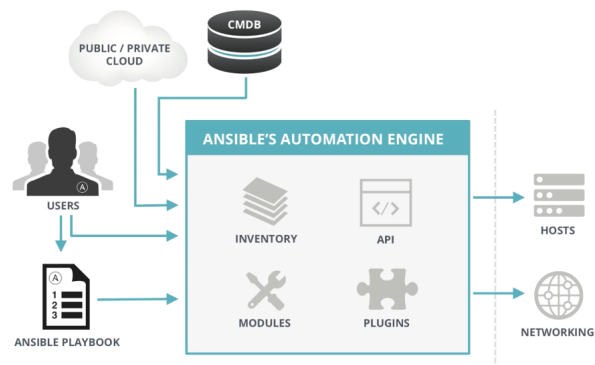
Inštalácia

Pre správu jednotlivých počítačov (uzlov) musíme nainštalovať na tzv. kontrolný uzol, ktorým je počítač, z ktorého bude prebiehať riadenie, program Ansible [12]. Výhodou je, že na žiaden spravovaný uzol nemusíme inštalovať agenta, ktorý by komunikoval a bol riadený kontrolným uzlom, pretože na systémoch GNU/Linux sa na riadenie použije SSH. Tento dizajn zjednodušuje prvotné nasadenie samotného Ansible a zvyšuje bezpečnosť, pretože neexistuje agent, ktorý by mohol byť napadnutý potenciálnym útočníkom.

Kľúčové prvky

Na automatizovanie nasadenia, úloh a správu je potrebných niekoľko kľúčových prvkov Ansible. Na obrázku 2.4 môžeme vidieť architektúru Ansible spolu s kľúčovými prvkami.

- Ansible Playbook – využíva jazyk YAML a definuje úlohy a role pre jednotlivé uzly.
- Inventory – zoznam uzlov, ktoré sú dostupné Ansiblu. Je to konfiguračný súbor obsahujúci IP adresu a meno uzlu (hostname), ktorý môže byť riadený.
- Modules – jednotky, ktoré vykonávajú a riadia danú úlohu, umožňujú inštaláciu, volania API a prístup k systémovým súborom. Moduly môžu byť napísané v štandardných skriptovacích jazykoch, ako napríklad Python, Bash, Perl.



Obr. 2.4: Architektúra Ansible [12]

Ansible Playbook

Príklad Ansible Playbook môžeme vidieť nižšie 1. Súbor definujúci nasadenie je napísaný v jazyku YAML a vykonáva sa sekvenčne. Nižšie uvedený playbook obsahuje jeden play, tri úlohy (tasks) a jeden handler. Na riadku číslo 2 definujeme skupinu zo zoznamu uzlov Inventory. Na riadkoch 3–5 deklarujeme premenné, s ktorými budeme pracovať. Nasledujú úlohy (tasks), ktoré musia byť pomenované a ďalej na riadku 9 definujeme použitý modul, konkrétne yum a čo má vykonať. Taktiež je možné použiť šablónu (template), ako na riadku 11, ktorá obsahuje deklarácie premenných, ktoré prepíšu deklarované premenné v playbo-oku. Každý playbook môže obsahovať na konci maximálne jeden handler, ktorý v našom prípade reštartuje službu apache.

Algoritmus 1: Ansible Playbook

```

1 ---
2 - hosts: webservers
3   vars:
4     http_port: 80
5     max_clients: 200
6     remote_user: root
7   tasks:
8     - name: ensure apache is at the latest version
9       yum: name=httpd state=latest
10    - name: write the apache config file
11      template: src=/srv/httpd.j2 dest=/etc/httpd.conf
12    notify:
13      - restart apache
14    - name: ensure apache is running (and enable it at boot)
15      service: name=httpd state=started enabled=yes
16  handlers:
17    - name: restart apache
18      service: name=httpd state=restarted

```

2.10 Python Fabric

Okrem rozsiahleho počtu nasadzovacieho softvéru, existujú aj knižnice a frameworky pre jednotlivé skriptovacie jazyky. Takouto knižnicou je aj Python Fabric [4], ktorý zefektívňuje použitie SSH pre nasadenie softvéru a administráciu systému. Umožňuje tiež upload a download súborov na uzly a vzdialené spúšťanie úloh. Táto knižnica je určená pre Python skripty verzie 2.5–2.7, no existujú aj porty, ktoré podporujú verzie 3.x, ako napríklad Python Fabric3. Taktiež ako Ansible nepotrebuje žiadneho agenta na nasadzovanom počítači a vystačí si s prístupom k SSH, čo zvyšuje bezpečnosť celého riešenia.

Vybrané funkcie Python Fabric [4]:

- `local` – vykonanie lokálneho príkazu
- `run` – vykonanie príkazu na vzdialených špecifikovaných uzloch
- `sudo` – použitie príkazu `sudo` na vzdialených uzloch
- `put` – nakopírovanie lokálneho súboru na uzol
- `get` – stiahnutie súboru z uzlu do lokálneho zariadenia
- `prompt` – vyslanie požiadavku užívateľovi na textový vstup
- `reboot` – reštart vzdialeného uzlu

env

Slovník `env` v sebe uchováva dôležité premenné prostredia, jedny z mnohých sú uzly v premennej `env.hosts`, skupiny definujúce role v `env.roledefs`, užívateľov pod ktorými budú príkazy spúšťané, ich heslá a mnohé ďalšie. Mnohé z týchto premenných sa okrem deklarácie v skripte môžu zadať aj pri spustení Python Fabric ako parametre programu.

fab

Python Fabric môžeme spúšťať buď pomocou príkazu `fab` a vypísaním parametrov programu, alebo si môžeme vytvoriť tzv. `fabfile`. Takýto `fabfile` nám umožňuje detailne špecifikovať uzly, zaradenie uzlov do skupín, definovať viaceré akcie a úlohy, čo nie je pri použití príkazu `fab` iba s parametrami buď možné, alebo dobre čitateľné. Nižšie uvedený príklad `fabfile` zobrazuje viaceré konštrukcie, ktoré boli spomenuté v predchádzajúcich riadkoch.

V prvom rade je nutné importovať knižnicu Python Fabric. Na riadkoch 2–7 sú deklarované jednotlivé premenné zo slovníku `env`, taktiež môžeme vidieť zatriedenie uzlov do skupín v premennej `env.roledefs`. Samotné úlohy sa zapisujú podobne ako funkcie v jazyku Python, až na to, že skripty/operácie, ktoré sa vykonávajú na vzdialenom uzle musíme zadať do konštrukcie `run`, ako môžeme vidieť na riadku 9.

Algorithmus 2: Fabfile

```
1 from fabric.api import env, run
2 env.user = 'implicit_user'
3 env.hosts = ['www1', 'www2', 'www3', 'ns1', 'ns2']
4 env.roledefs = {
5     'web': ['www1', 'www2', 'www3'],
6     'dns': ['ns1', 'ns2']
7 }
8 def taskA():
9     run('ls')
10 def taskB():
11     run('whoami')
```

Kapitola 3

Monitorovanie systému

Počas činnosti počítačového softvéru na výpočtovom zariadení dochádza k rôznym zmenám dát, nepredvídateľným hardvérovým poruchám a abnormálnemu správaniu systému, ktoré musíme zaznamenávať, ukladať a následne analyzovať. Tieto dôležité informácie nám napomáhajú k zaisteniu kvality služieb, pričom sledujeme vyťaženie zdrojov počítača ako množstvo zabranej operačnej pamäte, vyťaženie procesoru, zaplnenie diskových polí, hodnoty S.M.A.R.T., IP adresy pripojených užívateľov a zariadení, teploty hardvérových súčastí, chybové správy a rôzne ďalšie informácie. Pre každý operačný systém existuje mnoho nástrojov na monitorovanie chodu systému a hardvéru. Keďže sa táto bakalárska práca zaoberá monitorovaním systému GNU/Linux, v nasledujúcich podkapitolách spomenieme monitorovací softvér pre tento operačný systém.

3.1 Textové utility na monitorovanie

Systém GNU/Linux môže byť používaný aj bez grafického rozhrania, čo môžeme vidieť prevažne pri nasadení operačného systému na serveroch. Tieto programy sa spúšťajú cez terminál a ich výstup môže byť zobrazený znova v terminále alebo bude presmerovaný do súboru. Zobrazovanie výsledkov príkazu na terminál sa zväčša používa pri jednorazovom spustení a naopak presmerovanie do súboru využijeme na automatickú analýzu za pomoci skriptu. Výhodou týchto nástrojov je fakt, že nezaťažujú hardvér pre vykreslenie grafického okna, taktiež jednoduchšie strojové spracovanie a pri vzdialenom spustení cez SSH nezaťažujeme nadmerne počítačovú sieť.

vmstat

Tento nástroj je veľmi nápomocný pri zisťovaní výkonu systému. Môže byť spustený v dvoch módoch, a to average mód a sample mód [7]. V druhom zmienenom móde bude nástroj zaznamenávať štatistiku po vopred definovanú dobu, čo využijeme pri skúmaní systému počas záťaže.

Zobrazované informácie [7]:

- CPU – vyťaženie procesoru jadrom, prerušeniami, užívateľskými aplikáciami, nečinnosťou.
- Systém – počet prerušení, počet prepnutí kontextu.

- Operačná pamäť – výpis voľnej a využitej pamäte.
- SWAP – počet kB prenesených z RAM do SWAP a naopak.
- Blokové procesy – počet procesov čakajúcich na dokončenie IO operácií.

top

Tento softvér zobrazuje veľké množstvo informácií, ktoré sa môžeme dozvedieť z mnohým menších nástrojov. Podobne ako utilita **ps** zobrazuje bežiace procesy, PID, užívateľa, ktorý proces spustil a čas spustenia [8]. Okrem týchto informácií zobrazuje veľkosť voľného priestoru v RAM a SWAP, aktuálne vyťaženie systému, využitie CPU a počet prihlásených užívateľov [8]. Veľkou výhodou tohto programu je, že zobrazuje informácie v reálnom čase a jeho výpis je personalizovateľný. Veľmi obľúbeným sa stalo rozšírenie tohto nástroja s názvom **htop**.

ps

Nástroj na výpis aktuálne bežiacich procesov na počítači, nevypisuje ich v reálnom čase, robí iba snímku stavu pri spustení programu [8]. Umožňuje zobraziť názov procesu, jeho PID, čas spustenia, užívateľa, ktorý spustil proces a mnohé ďalšie informácie.

mpstat

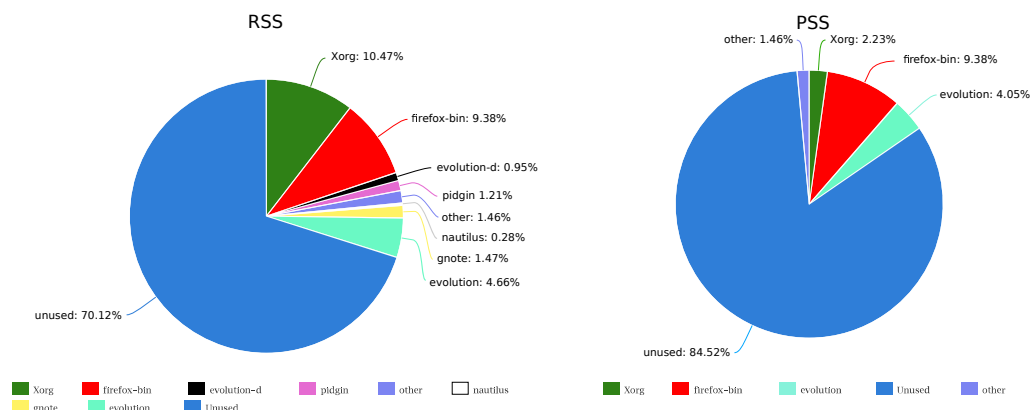
Nástroj slúži na detailné zobrazenie štatistiky procesoru a na rozdiel od **vmstat** poskytuje detailnejšie rozpisy využitia procesorového času. Navyše umožňuje výpis štatistiky pre jednotlivé jadrá, respektíve vlákna procesoru [7].

free

Nástroj, ktorým zisťujeme informácie o RAM a SWAP [8]. Zobrazuje maximálnu kapacitu oboch pamätí, voľnú a použitú kapacitu. Výhodou je jednoduchý textový výstup, ktorý sa dobre spracováva za pomoci skriptov.

smem

Podobne ako program **free** a **top** môžeme touto utilitou zistiť zabranú časť pamäte, no na rozdiel od programu **top** pre jednotlivé procesy zobrazuje **smem** presnejšie údaje o využití pamäti, pretože je optimalizovaná na využitie v systémoch, ktoré umožňujú zdieľať pamäťový priestor medzi viacerými aplikáciami a vláknami [13]. Využitú pamäť RAM jednotlivých aplikácií možno odčítať zo stĺpca **PSS**, v ktorej nie sú zahrnuté zdieľané triedky, typicky knižnice, a teda zobrazuje skutočné množstvo pamäti zabrané programom [13]. Rozdiel v zabranej a voľnej RAM, kde **RSS** je štandardný ukazovateľ vyťaženia RAM známy z aplikácie **top** a **PSS** je ukazovateľ zabranej časti pamäti bez zdieľaných knižníc môžeme vidieť na grafe 3.1. Ide o celkom signifikantný rozdiel, ktorý môže pri monitorovaní clusterov znamenať deliacu hranicu medzi dostatkom a nedostatkom operačnej pamäti.



Obr. 3.1: Rozdielny pohľad na vyťaženie RAM [13]

smartctl

Užitočný nástroj monitorujúci zdravie fyzických diskov či už rotačných alebo nepohyblivých flash diskov. K zisteniu stavu disku využíva monitorovací systém S.M.A.R.T., ktorý detekuje a posieľa správy o rôznych ukazateľoch spoľahlivosti v snahe predvídať zlyhania. Každá hodnota S.M.A.R.T. má svoj identifikátor, aktuálnu hodnotu a najhoršiu dosiahnutú hodnotu. Umožňuje zistiť počet zapnutí diskov, problém so sektormi, problém s radičom, teplotu disku a iné. [8]

ip

Týmto nástrojom môžeme zistiť aktuálnu konfiguráciu sieťových rozhraní, taktiež umožňuje vykonávať zmeny na rozhraniach, ako napríklad zmena IP adresy, brány a MAC adresy [7]. Z výpisu môžeme tiež vyčítať počet prijatých a odoslaných paketov na rozhraniach. Tento nástroj nahrádza zaužívaný program `ifconfig`.

blkid a lsblk

Tieto dva programy umožňujú zobraziť informácie o pripojených blokových zariadeniach, a to predovšetkým ich umiestnenie v adresári `/dev`, ich univerzálny identifikátor UUID, label zariadenia, typ súborového systému, prípadne číslo partície [7].

lsusb a lspci

Program `lsusb` slúži k zisteniu pripojených USB zariadení pričom o nich poskytuje informácie ako identifikačné číslo výrobcu zariadenia, modelu a jeho názov. Na zistenie zariadení pripojených cez zbernicu PCI a jej novších derivátov slúži program `lspci`. [7]

df

Pre jednoduché zistenie množstva zabraného priestoru na sekundárnych a terciárnych pamäťových zariadeniach môžeme použiť textovú utilitu `df` [8].

3.2 Syslog

V dnešnom svete plnom počítačov, sieťových prvkov a počítačových periférií vzniká veľké množstvo záznamov o ich činnosti, ktoré majú informatívny, respektíve ladiaci charakter alebo oznamujú rôzne poruchy a chyby pri činnosti softvéru a hardvéru. Tieto informácie je často nutné zaznamenávať nielen lokálne na zariadeniach, ale aj posilať počítačovou sieťou na centralizované miesto určené na analýzu týchto informácií. Z tohto dôvodu vznikol štandard syslog, ktorý zaznamenáva programové správy a umožňuje ich posilať po počítačovej sieti na syslog server.

Protokol syslog je typu klient/server, pričom úlohu klienta zastáva zariadenie, ktoré generuje logovacie správy a posila ich na server, na ktorom beží syslog démon [16]. Týmto syslog démonom alebo serverom môže byť aj zariadenie generujúce samotné syslog správy. Na prenos po počítačovej sieti využíva protokoly transportnej vrstvy TCP a UDP, pričom je možné posielané dáta zabezpečiť pomocou SSL/TLS [1], keďže neraz obsahujú citlivé informácie.

Pre správne fungovanie rôznych implementácií syslog musíme mať na všetkých zariadeniach správne nastavený čas [16]. Na presné nastavenie času a zároveň jeho pravidelnú synchronizáciu musíme na počítačoch alebo sieťových prvkoch nastaviť NTP server. Aktuálny čas pomáha k lepšiemu zisteniu poruchy a umožňuje odhaliť sériu závislých chýb na viacerých zariadeniach.

Súčasťou generovaných správ syslog je okrem času, kedy udalosť nastala taktiež zariadenie (Facility), úroveň (Severity level) a dodatočné textové informácie. Každé zariadenie má unikátny kód a kľúčové slovo, ktoré špecifikuje typ programu v syslog správe. Zároveň je pri každej správe definovaná úroveň, respektíve priorita, ktorá udáva mieru závažnosti informácie. [16] [1]

Typy priorít zostupne [1]:

- Emergency
- Alert
- Critical
- Error
- Warning
- Notice
- Info
- Debug

Príklady zariadení [16]:

- auth – správy súvisiace s bezpečnosťou
- authpriv – správy riadenia prístupu
- cron – správy generované cronom/plánovačom

- daemon – správy démonov a procesov
- kern – správy jadra systému
- mail – správy súvisiace s elektronickou poštou

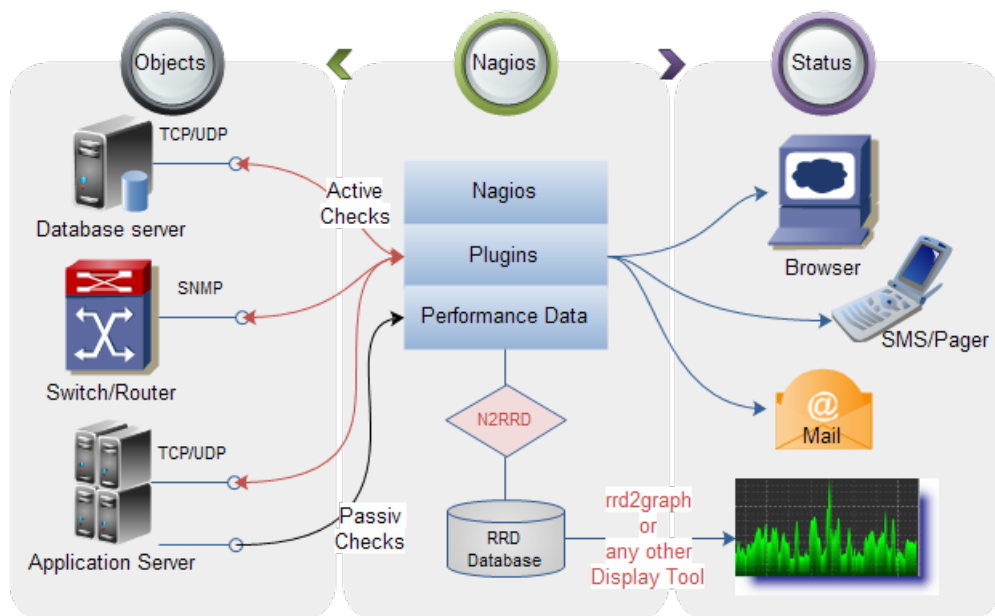
3.3 Nagios

Nagios je populárny open source program na sledovanie systému, počítačovej siete a infraštruktúry [10]. Primárne je vyvíjaný pre operačné systémy Linux, Unix a Unix – like. Od jeho vzniku v roku 2002 si získal veľký trhový podiel medzi monitorovacím softvérom najmä vďaka jeho bezplatnosti, použitiu licencie GNU GPL, ale hlavne dobrej podpore a množstvu oficiálnych a neoficiálnych rozšírení.

Vďaka rozsiahlej dokumentácii a dobrej podpore je inštalácia tohto monitorovacieho programu veľmi jednoduchá, no na správne fungovanie je potrebná rozsiahla konfigurácia, ktorá zaberá množstvo času. Diagram činnosti systému Nagios môžeme vidieť na obrázku č. 3.2.

Po nainštalovaní a nakonfigurovaní je dostupné grafické užívateľské rozhranie cez webový prehliadač [10]. Toto užívateľské rozhranie je personalizovateľné a zobrazuje rôzne dashbordsy s informáciami o stave siete, počte a druhu upozornení, vyťaženie zdrojov staníc a množstvo ďalších informácií.

Medzi nevýhody tohto softvéru patrí zastaralé používateľské rozhranie, neľahká konfigurácia a nutnosť serveru s veľkým množstvom pamäte.



Obr. 3.2: Princíp činnosti Nagios¹

¹<http://www.linux-magazin.de/news/nagios-plugins-in-version-2-0/>

3.4 Zabbix

Zabbix patrí k populárnym real-time nástrojom na monitorovanie počítačovej siete a aplikácií [20]. Tak ako nástroj Nagios je viac zameraný na monitorovanie počítačovej siete, no zvláda taktiež sledovanie serverov a služieb na nich bežiacich. Veľkou devízou Zabbixu oproti Nagios bola možná konfigurácia pomocou grafického rozhrania, čo však už dnes neplatí, pretože ju už oba nástroje podporujú.

Na ukladanie dát používa Zabbix relačné databáze, ako napríklad MySQL, PostgreSQL alebo Oracle SQL, je napísaný v programovacom jazyku C a na jeho frontend je využitý jazyk PHP [20].

Zabbix je možné používať ako na GNU/Linux, tak aj na systémoch Windows. Umožňuje sledovať služby ako SMTP, HTTP a iné, vyťaženie CPU, diskov a siete. Naproti systému Nagios ponúka modernejšie a intuitívnejšie grafické rozhranie a jednoduchšiu konfiguráciu.

Kapitola 4

Návrh

4.1 Požiadavky na aplikáciu

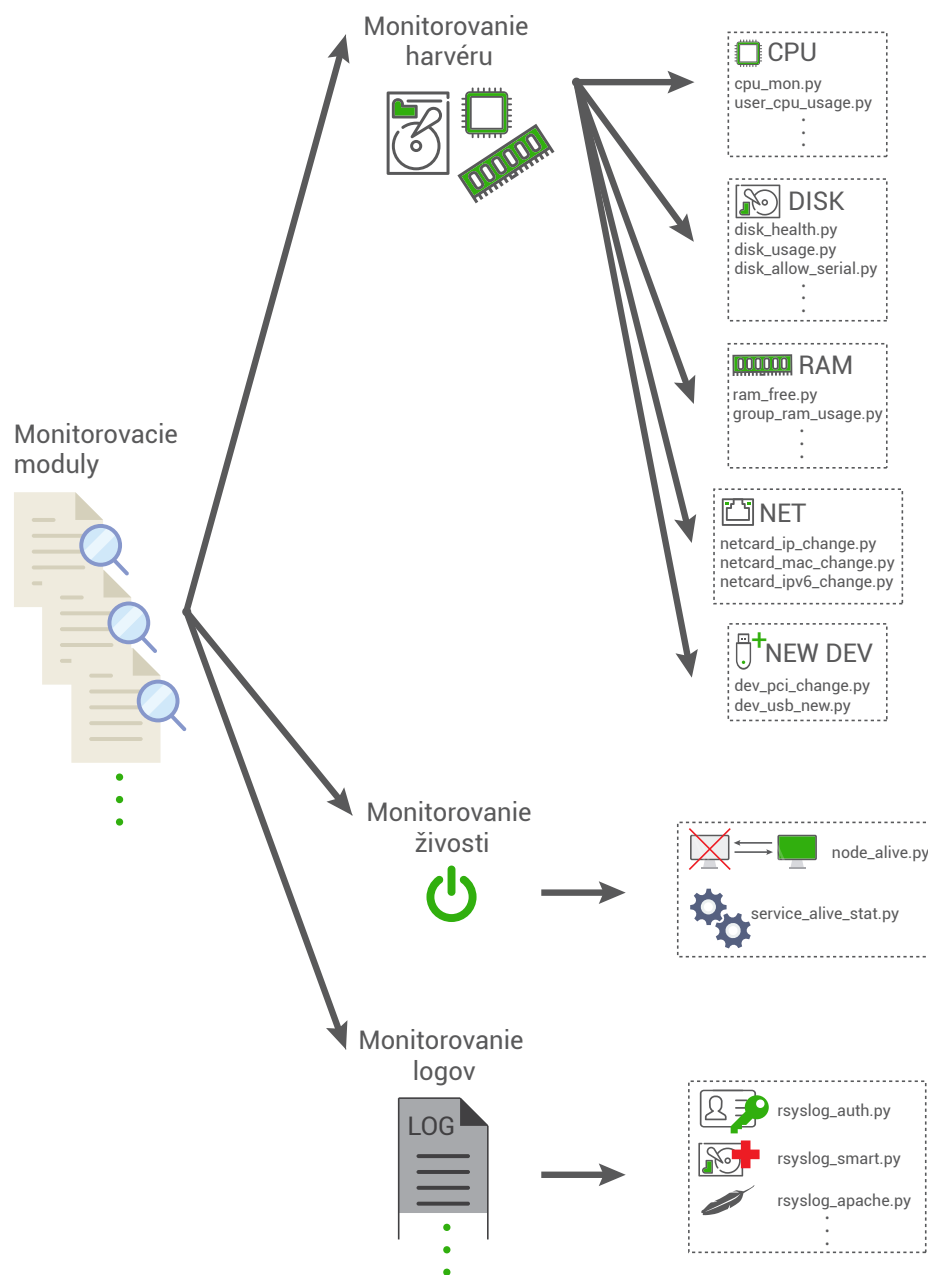
Kľúčovými vlastnosťami výsledného programu na monitorovanie uzlov v clusteri sú nenáročnosť z pohľadu výkonu a množstva prenesených dát po sieti, možnosť spustenia na rôznych GNU/Linux distribúciách, modulárnosť, jednoduchosť vytvorenia nových skriptov a integrácia už existujúcich skriptov.

Prenositeľnosť naprieč rôznymi distribúciami GNU/Linux je taktiež dôležitou požiadavkou, preto bol na implementáciu vybraný jazyk Python, ktorý okrem bezproblémového chodu na všetkých distribúciách je dobre editovateľný a čitateľný pre budúcich používateľov, ktorí si budú môcť pridávať vlastné skripty a meniť správanie aplikácie.

Samotný monitorovací nástroj bude rozdelený do piatich častí, a to:

- Inštalačný skript – `install.py`
- Riadiaci monitorovací a notifikačný skript – `linmon.py`
- Monitorovacie moduly v zložke `monitoring_scripts`
- Moduly monitorujúce rsyslog záznamy v zložke `syslog_scripts`
- Pomocné skripty v zložke `aux_scripts`

Monitorovací program je hierarchicky rozdelený z dôvodu ľahšej implementácie nových skriptov a tiež kvôli využitiu skriptov ako samostatné programy. Zoznam skriptov a ich zverených úloh sú popísané v prílohe [A](#).



Obr. 4.1: Hierarchické rozdelenie monitorovacích modulov

Pre bezproblémovú integráciu nových skriptov do monitorovacieho nástroja bude vypracovaná šablóna s manuálom, ako vytvárať skripty pre bezproblémovú funkcionálnu v monitorovacom programe a to najmä definícia výstupov nutná pre prácu hlavného skriptu.

Ako môžeme vidieť v prílohe A, administrátor systému je vždy oboznámený so zmenou stavu alebo chybou, ktorá nastala. Na tento účel bude použitá notifikácia pomocou emailu, kde bude okrem informácie, že skript zlyhal aj detailný výstup.

4.2 Odlišnosti oproti existujúcim riešeniam

Monitorovací softvér ako Nagios a Zabbix opísaný v kapitole 3 majú nesporné výhody v podpore a širokej nasadenosti. No pre niektoré vlastnosti je na monitorovanie zariadení nevhodný, ťažkopádny a zbytočne komplexný. V prvom rade monitorovací nástroj Linmon nepotrebuje databázy na uchovania monitorovacích informácií.

Častokrát výrobca alebo predajca zariadenia, ktoré potrebujú administrátori monitorovať dodáva svoje proprietárne monitorovacie riešenia a neumožňuje inštaláciu iných monitorovacích programov alebo umožňuje za cenu straty záruky. Pri týchto obmedzeniach však často umožňuje spúšťanie Python skriptov. Z tohto dôvodu je nástroj Linmon vhodnejším kandidátom na monitorovanie ako Nagios a Zabbix.

Zároveň umožňuje jednotné rozhranie pre monitorovacie skripty a zoskupenie notifikačných správ pri nájdení abnormálnych hodnôt, čo neumožňujú separátne úlohy v plánovači **cron**. Súčasťou takmer každej distribúcie GNU/Linux je interpret jazyku Python a program na prenos správ elektronickej pošty **sendmail**, ktoré sú zároveň jedinými nutnými prerekvizitami na chod nástroja.

V neposlednom rade veľkou výhodou je aj minimalistickosť a nenáročnosť na strane príjemcu notifikačných správ. Nagios a Zabbix potrebujú pre zobrazenie informácií webserver, no nástroju Linmon postačuje email klient na akomkoľvek zariadení alebo dokonca iba unix terminál, na ktorom je taktiež možné správy zobraziť.

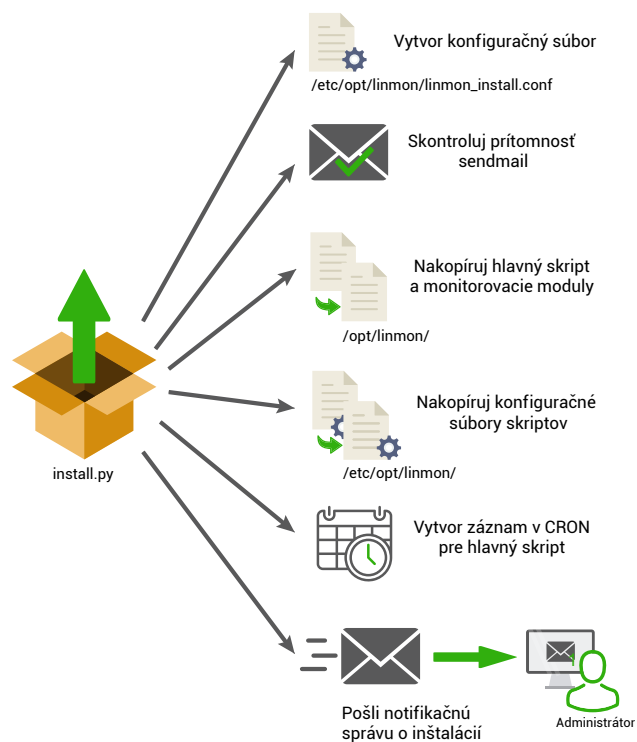
4.3 Inštalácia

Inštalčný skript je zodpovedný za zavedenie monitorovacieho nástroja do systému, čo zahŕňa uloženie konfiguračných dát pre zariadenie, nakopírovanie monitorovacích modulov a ich konfiguračných súborov do systému a taktiež je ním možné monitorovací nástroj odinštalovať. Prácu tohto skriptu ilustruje vývojový diagram 4.3.

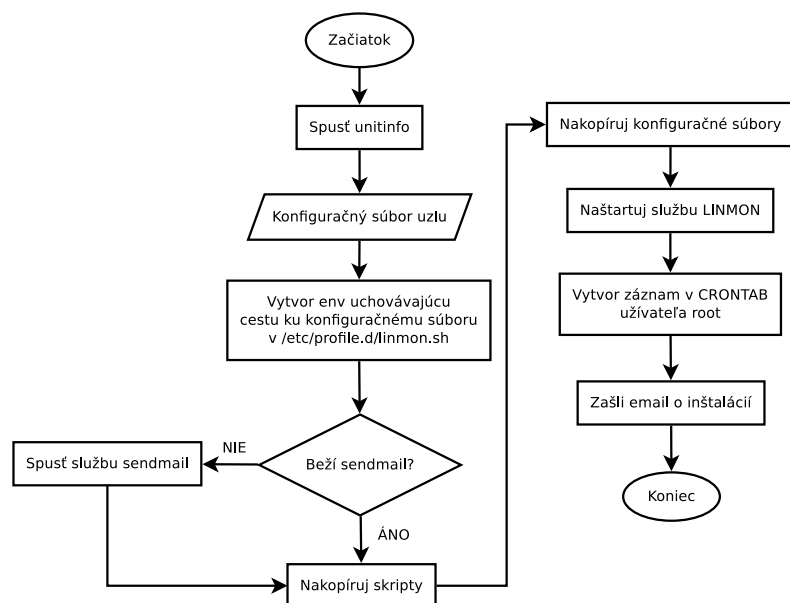
Podľa konvencií hierarchie súborového systému má byť pre vlastné binárne súbory a skripty použitý priečinok **/opt** a pre ich konfiguračné súbory **/etc/opt** [5]. Preto bude ako predvolená inštalčná zložka nástroja Linmon **/opt/linmon/** a ich konfiguračné súbory budú pri inštalácii zavedené do **/etc/opt/linmon/**.

Z dôvodu uchovania nastavení zadaných v príkazovom riadku pri inštalácii je vytvorený konfiguračný súbor uzlu obsahujúci zoznam emailových adries určených na notifikáciu, cestu k zložke so skriptami a ku konfiguračným súborom a cestu k binárnemu súboru **sendmail**.

Na záver inštalácie je zaslaný informačný email, v ktorom je hlavička, ktorá jednoznačne identifikuje zariadenie, na ktorom inštalácia prebehla a taktiež informácia o úspešnosti inštalácie s prípadnými chybami.



Obr. 4.2: Zjednodušený pohľad na inštaláciu skriptu



Obr. 4.3: Diagram počiatočnej inštalácie a konfigurácie skriptu

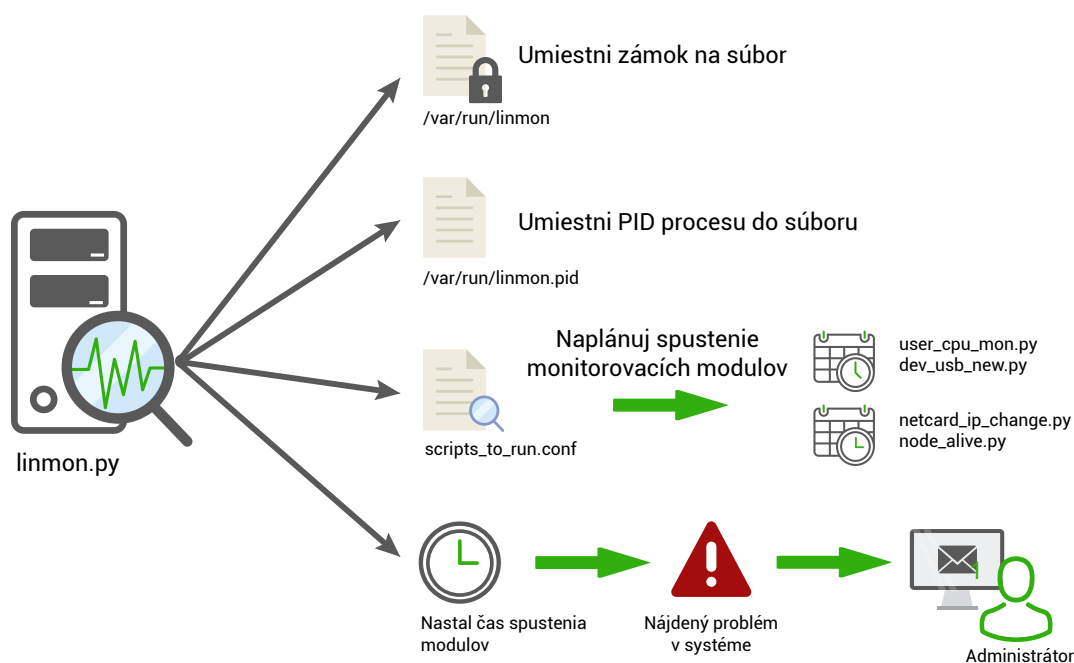
4.4 Hlavný skript

Hlavný skript `linmon.py` sa stará o spúšťanie definovaných skriptov (monitorovacích modulov) a následnú notifikáciu administrátora pri nájdení abnormálnych hodnôt monitorovacími modulmi.

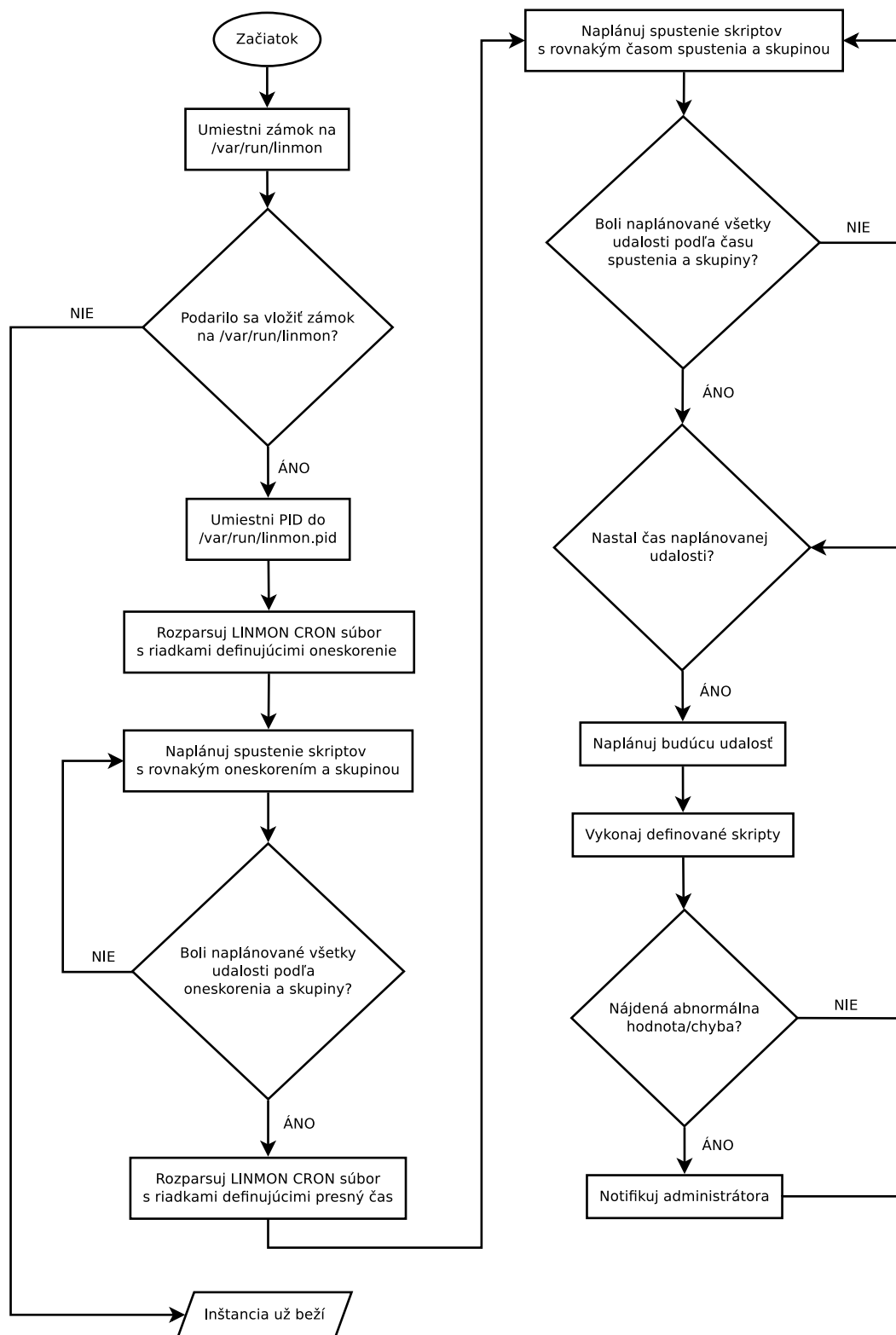
Dôležitým faktorom pri monitorovaní počítačov je zaistenie stáleho behu monitorovacieho skriptu. Táto požiadavka bude zaistená vložением skriptu do plánovača `cron`, kde sa každých 5 minút overí chod skriptu a taktiež za pomoci `init` systému. Keďže sa každých 5 minút spustí program, treba zaistiť, že pokiaľ je už program spustený, tak nespustí monitorovacie moduly, pretože by mohlo prísť k opakovanej notifikácii. Takéto správanie sa docieľi umiestnením zámku na súbor `/var/run/linmon`.

Jednotlivé skripty popísané v prílohe A budú spúšťané z hlavného skriptu v pravidelných intervaloch, kde každý skript môže mať tento interval odlišný.

Po počiatočnom načítaní a naplánovaní spustenia monitorovacích modulov sa skript dostane do nekonečného cyklu, kde sa na základe času spúšťajú definované skripty, naplánujú sa ich ďalšie spustenia a v prípade nájdenia abnormálnych hodnôt príde k notifikácii administrátora.



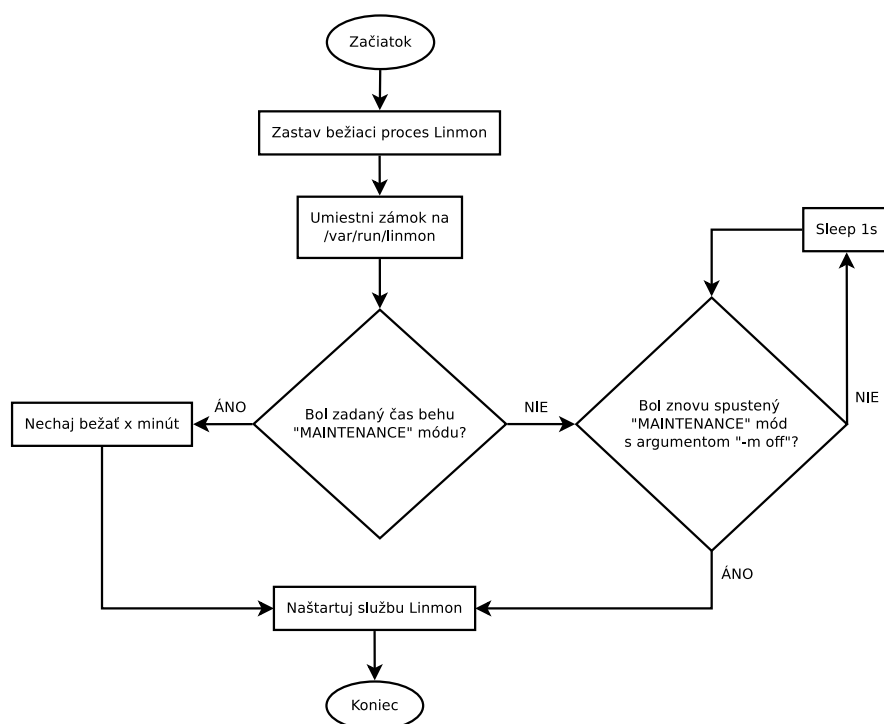
Obr. 4.4: Zjednodušený pohľad na fungovanie hlavného skriptu



Obr. 4.5: Vývojový diagram hlavného skriptu

Maintenance mód

Monitorovací skript bude spustený ako démon, teda bude bežať na pozadí a v pravidelných intervaloch spúšťať monitorovacie moduly. No, ak počas monitorovania systému dôjde k servisnému zásahu administrátora na monitorovanom zariadení, môže to vyvolať reťaz notifikácií a zahltenie poštovej schránky prijímajúcej notifikácie. Z tohto dôvodu bude v hlavnom skripte implementovaný tzv. maintenance mód, ktorým sa dočasne vypne monitorovanie a zasielanie správ o chybách. Vývojový diagram 4.6 prezentuje činnosť tejto funkcie, pričom ako môžeme vidieť, je možné tento mód zapnúť a následne vypnúť po ukončení údržby alebo špecifikovať dobu, po ktorú bude bežať.



Obr. 4.6: Vývojový diagram módu údržby

Kapitola 5

Implementácia

V tejto kapitole sa bližšie pozrieme na zaujímave pasáže a kľúčové prvky implementácie monitorovacieho a notifikačného programu `linmon.py`, taktiež si rozoberieme implementáciu monitorovacích modulov starajúcich sa o sledovanie jednotlivých služieb alebo zdrojov výpočtového zariadenia.

Ako implementačný jazyk pre túto bakalársku prácu bol vybraný skriptovací jazyk Python, ktorý je zväčša dostupný v každej distribúcii GNU/Linux alebo je ho možné veľmi jednoducho doinštalovať. Na zaistenie čo najmenšieho zásahu do operačného systému, teda inštalácie a z dôvodu zabránenia nekompatibility a konca vývoja rôznych knižníc boli v implementácii použité iba vstavané knižnice jazyka Python dostupné od verzie 3.5.

5.1 Hlavný skript Linmon

Hlavný skript Linmon riadi jednotlivé monitorovacie moduly monitorujúce či už záznamy `rsyslog` alebo služby a zdroje. Je kompatibilný so súčasnými init systémami ako sú `SystemD`, `SysV`, `Upstart`. Nepretržitý chod skriptu je zabezpečený dvoma na sebe nezávislými spôsobmi popísanými v podkapitole Zaistenie vysokej dostupnosti.

Implementácia vlastného plánovača

Ako už bolo spomenuté, skript Linmon iba spúšťa jednotlivé definované monitorovacie moduly, ktoré sa zameriavajú na jednu špecifickú oblasť monitorovania. Tieto moduly musia byť niekde vopred špecifikované, aby mal skript povedomie o jednotlivých exekúciách. V prvej fáze vývoja, respektíve návrhu aplikácie sa počítalo s využitím vstavaného linuxového plánovača `cron`, od tohto smerovania bolo však upustené z viacerých dôvodov, a to:

- Skripty s rovnakým naplánovaným časom spustenia by nemali údaje na výstupe v rovnakom čase a nebolo by možné agregovať skripty naplánované v rovnakom čase do jedného notifikačného emailu.
- Nemožnosť vytvorenia zoskupenia skriptov vrámci rovnakého času spustenia, teda pokiaľ by sme chceli v rámci skriptov spustených každých 5 minút dostať jednu agregovanú správu, ale rozdeliť v rámci týchto piatich minút správy na dve alebo viac na základe nejakej značky.

- Nemenej dôležitou nevýhodou je zmiešanie už existujúcich úloh v plánovači `cron` s úlohami pre tento nástroj, čo by viedlo aj ku komplikovanému odstraňovaniu pri odinštalovaní skriptu a neprehľadnosti pri údržbe.

Algoritmus 3: `scripts_to_run.conf`

```

1 5 ./user_cpu_usage.py root -t 10 -i info
2 5 ./disk_temp.py -a -t 30 -i warning
3 5 ./disks_smart_mon.py -i critical
4 5 [AUTH] ./rsyslog_auth.py -i critical
5 5 [RAM] ./ram_free_mon.py -p 20 -i critical
6 5 [RAM] ./ram_free_mon.py -p 40 -i error
7 5 [RAM] ./ram_free_mon.py -p 50 -i warning
8 5 [RAM] ./ram_free_mon.py -p 90 -i info
9 10 ./proc_ram_mon.py -p 20 apache -i warning
10 10 ./proc_cpu_mon.py -p 15 apache -i warning
11 8:30 [ARRAYS] ./disk_health_mdadm.py -a -i error
12 8:30 [ARRAYS] ./disk_health_hp.py -a -i error
13 10:00 ./disk_temp_hp.py -a -i warning

```

Vyššie uvedený konfiguračný súbor **3** sa používa na definovanie spúšťaných skriptov a ako je možné vidieť, jeho syntax je inšpirovaná linuxovým plánovačom `cron`. Oproti vstavanému plánovaču je o poznanie jednoduchší jednak z dôvodu jednoduchšej implementácie, ale aj z dôvodu nadbytočnosti prípadov použitia pre monitorovanie.

Každý riadok konfiguračného súboru definujúceho spúšťané skripty sa skladá z dvoch respektíve troch častí oddelených medzerou. Prvou časťou je čas exekúcie skriptov, ktorý sa dá špecifikovať dvoma spôsobmi, a to definovaním času v minútach medzi jednotlivými spusteniami, takýmto príkladom sú riadky 1–10 v konfiguračnom súbore **3**. Druhým možným zápisom je definovanie presného času spustenia na riadkoch 11–13 v konfiguračnom súbore **3**, pričom takto naplánované skripty budú spustené iba raz za deň v špecifikovaný čas.

Druhá časť je voliteľná pričom umožňuje rozdelenie na dve alebo viacej notifikačných správ na základe značky v hranatých zátvorkách. Pre lepšie pochopenie si tento prístup vysvetlíme na príklade konfiguračného súboru `scripts_to_run.conf` **3**. Na riadkoch 1–8 máme skripty, ktoré budú spustené každých 5 minút, znamenalo by to, že ak by prišlo k abnormálnej hodnote, príde administrátorovi jeden email, v ktorom bude záznam s výstupmi týchto šiestich skriptov. V našom prípade však máme na riadku 4 a na riadkoch 5–8 definované dve rozdielne značky, ktoré zapríčinia, že v prípade nájdenia abnormálneho hodnotu príde tri emaily namiesto jedného, čo umožňuje lepšie filtrovanie notifikačných správ. Teda z hľadiska času exekúcie sa skripty na riadkoch 1–8 správajú totožne, t.j. sú spustené s rovnakým oneskorením, no z hľadiska notifikácie sa správajú nezávisle na základe definovaných značiek. V prípade, že nie je definovaná značka v hranatých zátvorkách napr. riadky 1–3, prípadne 9–10, tak sa v notifikačnej správe použije značka `DEFAULT`.

Posledná časť obsahuje cestu k skriptu, ktorý sa má spustiť aj s jeho definovanými parametrami spustenia.

Časová jednotka	Linux Cron	Linmon Cron
Každých 5 minút	* / 5 * * * *	5
Raz denne v 12:00	0 12 * * *	12:00

Tabuľka 5.1: Ekvivalencia zápisov plánovačov

Formát notifikačnej správy

Jedným z hlavných dôvodov, prečo tento monitorovací nástroj vznikol je nielen možnosť zoskupovať správy monitorujúcich skriptov a tým zvýšiť prehľadnosť, ale aj jednoduché filtrovanie správ a zjednotenie výstupov monitorovacích modulov.

Notifikačná správa sa skladá z častí ako predmet správy, hlavička správy a telo správy. Telo správy sa odlišuje v závislosti od toho, či ide o správu o inštalácii skriptu alebo o správu obsahujúcu informácie o nájdených abnormálnych hodnotách na monitorovanom zariadení.

```

From Juraj Korček <linmon.relay@gmail.com> ☆
Subject LINMON installation report from thinkpadl460

Hostname: thinkpadl460
Distribution: Debian GNU/Linux buster/sid
Kernel version: 4.15.0-2-amd64
Init system type: systemd
Package manager: apt
Network adapter "UP": enp0s31f6 192.168.3.5/24
Network adapter "UP": wlp3s0 192.168.3.3/24
Config file: /etc/opt/linmon/linmon_install.conf
Config env file: /etc/profile.d/linmon.sh

-----
Installation checklist
-----
Sendmail running: OK
Init system file enabled: OK
Init system file started: OK
Created cron: OK

-----
Installation errors
-----
Sendmail error:
Init system file error(s):
Created cron error(s):

```

Obr. 5.1: Notifikačná správa o inštalácii skriptu

```

From root <linmon.relay@gmail.com> ☆
Subject LINMON [error][5min][RAM] 3 problem(s) found

Hostname: thinkpadl460
Distribution: Debian GNU/Linux buster/sid
Kernel version: 4.15.0-2-amd64
Init system type: systemd
Package manager: apt
Network adapter "UP": enp0s31f6 192.168.3.5/24
Network adapter "UP": wlp3s0 192.168.3.3/24

Executed scripts status
-----
No.  Script                                                                                               Status
1.   /opt/linmon/monitoring_scripts/ram_free_mon.py -p 20 -i critical OK
2.   /opt/linmon/monitoring_scripts/ram_free_mon.py -p 40 -i error  ERR
3.   /opt/linmon/monitoring_scripts/ram_free_mon.py -p 50 -i warning ERR
4.   /opt/linmon/monitoring_scripts/ram_free_mon.py -p 90 -i info   ERR

-----ERROR script no.2-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 40 -i error
Executed time: 2018-04-11 12:20:57

THRESHOLD FREE RAM: 40%
CURRENT FREE RAM: 34.55822750607943%
Mem:      total      used      free      shared  buff/cache  available
Swap:     8195       1857      6338      1098     2289       1259

-----END OF ERROR script no.2-----

-----WARNING script no.3-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 50 -i warning
Executed time: 2018-04-11 12:20:57

THRESHOLD FREE RAM: 50%
CURRENT FREE RAM: 34.571737368278846%
Mem:      total      used      free      shared  buff/cache  available
Swap:     8195       1857      6338      1098     2289       1260

-----END OF WARNING script no.3-----

-----INFO script no.4-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 90 -i info
Executed time: 2018-04-11 12:20:57

THRESHOLD FREE RAM: 90%
CURRENT FREE RAM: 34.55822750607943%
Mem:      total      used      free      shared  buff/cache  available
Swap:     8195       1857      6338      1098     2289       1260

-----END OF INFO script no.4-----

```

Obr. 5.2: Notifikačná správa pri nájdení abnormálnych hodnôt

Ako je možné vidieť na predchádzajúcich dvoch obrázkoch 5.1 a 5.2, tak hlavička informujúca o zariadení, na ktorom je skript nainštalovaný je prítomná ako v notifikačnej správe o inštalovaní, tak aj v správe informujúcej o nájdení abnormálnych hodnôt.

Význam a vysvetlenie položiek v notificačnej správe

Všetky nižšie popísané položky správy vychádzajú z obrázku 5.2 a z riadkov 5–8 konfiguračného súboru 3.

Na obrázku 5.3 je predmet správy pri nájdení abnormálneho správania systému. Skladá sa z uvádzacieho slova **LINMON**, ktorým začínajú všetky notificačné správy. Nasleduje kľúčové slovo s úrovňou závažnosti správy, ktoré sa definuje pri každom skripte ako parameter, pričom pokiaľ viac ako jeden skript nájde abnormálnu hodnotu, do predmetu sa vždy zapíše najzávažnejšia úroveň. Pokiaľ nie je úroveň pri skripte definovaná, tak sa použije najnižšia úroveň, t.j. **info**. Ďalšou položkou je čas, respektíve doba medzi jednotlivými spusteniami. Poslednou časťou napomáhajúcou filtrovaniu správ je značka, ktorá zapríčiní rozdelenie výstupov skriptov naplánovaných na rovnaký čas do viacerých správ. Skripty s rovnakou značkou sú súčasťou jednej správy, pokiaľ nie je táto značka definovaná a nie je treba oddeliť výstupy skriptov do viacerých správ, tak táto značka nadobudne hodnotu **DEFAULT**.

```
Subject: LINMON[thinkpad1460][error][5min][RAM]
```

Obr. 5.3: Predmet notificačnej správy pri nájdení abnormálnych hodnôt

Každá správa bez ohľadu na to, či notifikuje o inštalácii alebo o nájdení abnormálnej hodnote obsahuje hlavičku s popisom zariadenia, z ktorého bola odoslaná. Túto hlavičku reprezentuje obrázok 5.4, pričom zobrazuje základné informácie potrebné na indentifikáciu zariadenia, v ktorom prišlo ku chybe.

```
Hostname: thinkpad1460
Distribution: Debian GNU/Linux buster/sid
Kernel version: 4.15.0-2-amd64
Init system type: systemd
Package manager: apt
Network adapter "UP": wlp3s0 192.168.3.3/24
```

Obr. 5.4: Hlavička správ

Informačne najvýpovednejšou časťou správy je jej posledná časť správy 5.5, pretože obsahuje zoznam spustených skriptov v naplánovanú dobu a stav indikujúci nájdenie problému v systéme. Pokiaľ sa nájde problém, stav skriptu je označený kľúčovým slovom **ERR** a zároveň je vypísaný reťazec s bližšími informáciami o chybe. Ako je možné vidieť, každý jeden vypísaný reťazec začína s úrovňou závažnosti a sekvenčným číslom skriptu, ktorý našiel abnormálnu hodnotu.

```

Executed scripts status
-----
No.  Script                                                                                               Status
1.  /opt/linmon/monitoring_scripts/ram_free_mon.py -p 20 -i critical      OK
2.  /opt/linmon/monitoring_scripts/ram_free_mon.py -p 40 -i error        ERR
3.  /opt/linmon/monitoring_scripts/ram_free_mon.py -p 50 -i warning      ERR
4.  /opt/linmon/monitoring_scripts/ram_free_mon.py -p 90 -i info         ERR

-----ERROR script no.2-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 40 -i error
Executed time: 2018-04-25 08:35:01

THRESHOLD FREE RAM: 40%
CURRENT FREE RAM: 37.7124359320205%
      total      used      free      shared  buff/cache  available
Mem:      7414      4616      136      873      2660      1868
Swap:      8195       51     8144

-----END OF ERROR script no.2-----

-----WARNING script no.3-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 50 -i warning
Executed time: 2018-04-25 08:35:01

THRESHOLD FREE RAM: 50%
CURRENT FREE RAM: 37.7124359320205%
      total      used      free      shared  buff/cache  available
Mem:      7414      4617      135      873      2661      1867
Swap:      8195       51     8144

-----END OF WARNING script no.3-----

-----INFO script no.4-----
Script: /opt/linmon/monitoring_scripts/ram_free_mon.py -p 90 -i info
Executed time: 2018-04-25 08:35:01

THRESHOLD FREE RAM: 90%
CURRENT FREE RAM: 37.7124359320205%
      total      used      free      shared  buff/cache  available
Mem:      7414      4617      135      873      2661      1867
Swap:      8195       51     8144

-----END OF INFO script no.4-----

```

Obr. 5.5: Telo správy s výstupmi skriptov

Zaistenie vysokej dostupnosti

Dlhodobé výpadky monitorovacieho nástroja Linmon bez mechanizmu automatického znovu spustenia po chybe by znamenali absenciu dôležitých správ o stave systému. Preto je nutné sa výpadkom vyvarovať, no niekedy nemusia byť nutne zapríčinené chybou v aplikácií, ale rôznymi okolnosťami v operačnom systéme. Práve v prípade neočakávanej udalosti v systéme a následnom zastavení monitorovacieho skriptu Linmon je dôležité čo najskôr znovu spustiť tento nástroj. Mechanizmus na zaistenie neustáleho behu, respektíve znovuspustenia je implementovaný dvoma nezávislými spôsobmi.

Mechanizmy znovuspustenia:

- periodická úloha v plánovači **cron**
- nastavenia v **init** systéme umožňujúce znovu spustenie

Prvý zmienený spôsob je realizovaný nasledujúcou úlohou, ktorá každých 5 minút spúšťa monitorovací skript:

```
*/5 * * * * /opt/linmon/linmon.py
```

Druhým mechanizmom zaistujúcim opätovné zapnutie monitorovania je pomocou **init** systému. Každý z podporovaných **init** systémov má inú konvenciu zápisu a spôsob reštartu služby pri chybe.

Riešenia znovuspustenia v **init** systémoch:

- **SystemD** – pridanie položky **Restart=on-failure** do súboru **/etc/systemd/system/linmon.service**.
- **SysV** – pridanie záznamu **linm:2345:respawn:/opt/linmon/linmon.py** do súboru **/etc/inittab**.
- **Upstart** – pridanie záznamu **respawn** do súboru **/etc/init/linmon.conf**

Zaistenie chodu iba jednej inštancie programu

Program **linmon.py** nie je možné spustiť vo viacerých inštanciách, pretože takéto správanie by mohlo mať za následok opakovanie notifikačných správ a neúmerne zahľtenie poštovej schránky. K opätovnému spusteniu by mohlo prísť buď náhodným spustením alebo pravdepodobnejšie pomocou linuxového plánovača, ktorý periodicky spúšťa program.

Pri každom spustení **linmon.py** pokiaľ nie sú zadané argumenty na mód údržby alebo **update**, sa zavolá funkcia **lock_file_fn()** z modulu **linmon_builtin**. Pokiaľ sa pomocou **fcntl.lockf(file_lock, fcntl.LOCK_EX|fcntl.LOCK_NB)** nepodarí umiestniť zámok na súbor **/var/run/linmon**, znamená to, že už je skript spustený a nemôže byť spustená ďalšia inštancia.

Mód údržby

Mód údržby je dôležitou súčasťou monitorovacieho programu, pretože umožňuje ignorovať servisné a testovacie zásahy na zariadeniach, ktoré by mohli vyvolať nadmerné nežiadané notifikácie o chybách a abnormálnom správaní. Tento mód je možné spustiť prepínačom **-m on**, čo má za následok zavolanie funkcie **maintenance_mode**. V prvom kroku sa zavolá funkcia **proc_kill**, ktorá otvorí súbor **/var/run/linmon.pid** uchovávajúci PID bežiaceho procesu programu **Linmon** a následne sa tomuto procesu pošle signál **SIGKILL**. Pre zabránenie spustenia skriptu metódami zaistujúcimi stály chod skriptu je potreba ešte umiestniť zámok na súbor **/var/run/linmon** pomocou funkcie **lock_file_fn()**.

Po úspešnom umiestnení zámku je treba zistiť, v ktorom režime má mód údržby bežať, pokiaľ je zadán aj prepínač **-t** s definovaným časom, tak sa využije funkcia **sleep**. Po uplynutí administrátorom definovaného času, sa zavolá funkcia **init_script_start()**, ktorá spustí opätovné monitorovanie.

V opačnom prípade, keď nie je špecifikovaný čas, tak po vykonaní údržby stačí spustiť program `linmon.py` s prepínačom `-m off`, čo spôsobí opätovné spustenie monitorovania a zašle správu o vypnutí módu údržby.

Podporované init systémy

Implementácia programu bola robená s ohľadom na prenositeľnosť a kompatibilitu medzi viacerými distribúciami GNU/Linux. Aj keď dnes najpoužívanejším init systémom je SystemD, niektoré distribúcie spoliehajú stále na veľmi rozšírený SysV a Upstart. Preto sú všetky tieto tri init systémy podporované programom Linmon. Na zistenie použitého init systému slúži funkcia `init_system_fn()` z modulu `unitinfo`. Táto funkcia zistí, kam odkazuje symbolický odkaz procesu s číslom 1 a podľa toho rozhodne o nainštalovanom a používanom init systéme. Inštalčný skript na základe použitého init systému zavedie potrebné súbory init systému a vykoná ich spustenie.

Súbor `/var/run/linmon.pid` je potrebný nielen pre mód údržby, ale aj v prípade inti systému SysV, ktorý podľa PID v tomto súbore umožňuje zastaviť službu monitorovania.

Pri použití ktoréhokolvek init systému sa služba monitorovania Linmon spúšťa pod užívateľom `root`, čo je nevyhnutné hlavne pre monitorovacie moduly, ktoré často potrebujú plný prístup k systému.

Úrovně závažnosti

Pre potreby filtrovania dôležitosti správ zaslaných administrátorovi sa do správ vkladá jeden zo štyroch typov závažnosti. Táto závažnosť je definovaná pre každý monitorovací modul, no nie je v moduloch natvrdo definovaná, administrátor si ju môže podľa zaužívaných zvyklostí a role zariadenia modifikovať. Príklady vstupov boli prezentované v konfiguračnom súbore [3](#) a výstup je prezentovaný v obrázku [5.2](#).

Na výber sú tieto závažnosti (vzostupne):

- info
- warning
- error
- critical

5.2 Moduly monitorujúce zdroje a služby

Moduly, ktoré sa starajú o sledovanie zdrojov zariadenia a služieb sú skripty napísané v jazyku Python, no hlavný monitorovací skript `linmon.py` podporuje aj skripty napísané v jazyku Bash. De facto ide o jednoduché skripty, ktoré sledujú jednu konkrétnu hodnotu, prípadne výstup určitého programu.

Tieto skripty používajú zväčša vstavané textové programy ako napríklad `top`, `free`, `ip`, prípadne jednoducho doinštalovateľné programy `smem`, `mdadm` a iné. Z tohto dôvodu je na začiatku každého skriptu ihneď po shebangu ako komentár pridaný zoznam programov potrebných pre chod skriptu. Príklad takéhoto komentáru môžeme vidieť na riadku číslo 3 výseku zdrojového kódu [4](#).

Algoritmus 4:

```
1 #!/usr/bin/python3
2
3 #DEPENDENCIES: blkid,df
4
5 import argparse
```

Každý zo skriptov obsahuje dva rovnaké prepínače, sú nimi `-e` a `-i`, ostatné prepínače závisia od konkrétnej monitorovanej súčasti systému alebo zariadenia. Prepínač `-e` alebo v dlhom formáte `--execute` slúži na špecifikovanie akcie respektíve programu, ktorý sa spustí pri nájdení abnormálnej hodnoty alebo problému monitorovaným skriptom. Spustenie programu alebo špecifická akcia mohla byť natvrdo zadaná v zdrojovom kóde skriptu, no prepínač poskytuje vyššiu flexibilitu, pretože napríklad skript `ram_free_mon.py` môže mať definované rôzne prahové hodnoty a pri každej administrátor môže chcieť spustiť iný program alebo akciu a zároveň nie všetci administrátori chcú reagovať rovnakým spôsobom na abnormálne správanie systému. Spúšťanie skriptu zabezpečuje funkcia `script_action`.

Druhý prepínač spoločný pre každý skript `-i` alebo v dlhom formáte `--importance` určuje dôležitosť skriptu, ktorá bude premietnutá do výpisu a predmetu notifikačnej správy 5.2. Pri nájdení problému v systéme sa sformuje notifikačná správa za pomoci funkcie `print_output` pričom prvý riadok bude obsahovať dôležitosť, ktorá bola definovaná prepínačom a ďalej bude pokračovať detailný výpis skriptu v závislosti na monitorovanom zdroji alebo službe. V prípade nešpecifikovania dôležitosti, kde prvý riadok skriptu obsahuje už detailný výpis, tak hlavný skript `linmon.py` priradí predvolenú najnižšiu dôležitosť `info`. Pokiaľ sa nenájde abnormálna hodnota, skript vráti buď prázdny nový riadok alebo sa ukončí bez výpisu na štandardný výstup.

5.3 Moduly monitorujúce rsyslog záznamy

Skripty starajúce sa o sledovanie výpisov syslog dodržujú všetky princípy ako skripty spomenuté v podkapitole 5.2. Zároveň platí, že každý skript monitoruje špecifickú službu, ako napríklad chyby autentizácie, chyby v programoch na prijímanie a odosielanie elektronickej pošty, rôznych serverových programov atď.

Skript monitorujúci syslog záznam konkrétnej služby alebo programu potrebuje vždy prístup k nejakému súboru, ktorý ukladá tieto záznamy, no distribúcie majú rôzne konvencie, kde ukladať určité záznamy. Preto funkcia `installed_dist` zistí distribúciu nainštalovanú na zariadení a podľa toho určí, kde bude skript hľadať zvolenú chybu. Ak by tento prístup zlyhal je ešte možné špecifikovať prehľadávaný súbor pomocou prepínača `-f`.

Keďže sa skript spúšťa opakovane, tak je treba zaistiť, že súbor so záznamami syslog bude skúmaný od riadku, kde predchádzajúce skenovanie skončilo. Z tohto dôvodu je pri spustení skriptu vytvorený súbor v zložke `/tmp/linmon/` pre každý skript zvlášť, do ktorého je zapísané číslo riadku, od ktorého má začať nasledujúce skenovanie. Pri prvom spustení sa záznamy neskúmajú, zapíše sa aktuálne posledné číslo riadku, ktoré sa zistí funkciou `count_lines` a od neho sa pri druhom spustení začne skúmanie záznamov funkciou `examine_new_logs`.

Okrem skriptov, ktoré monitorujú konkrétne všeobecne známe chyby v záznamoch rsyslog, existuje skript `rsyslog_universal.py`, ktorý po zadaní súboru v ktorom má prehľadávať a regulárneho výrazu, ktorý sa porovná s riadkami záznamov umožňuje prehľadávať

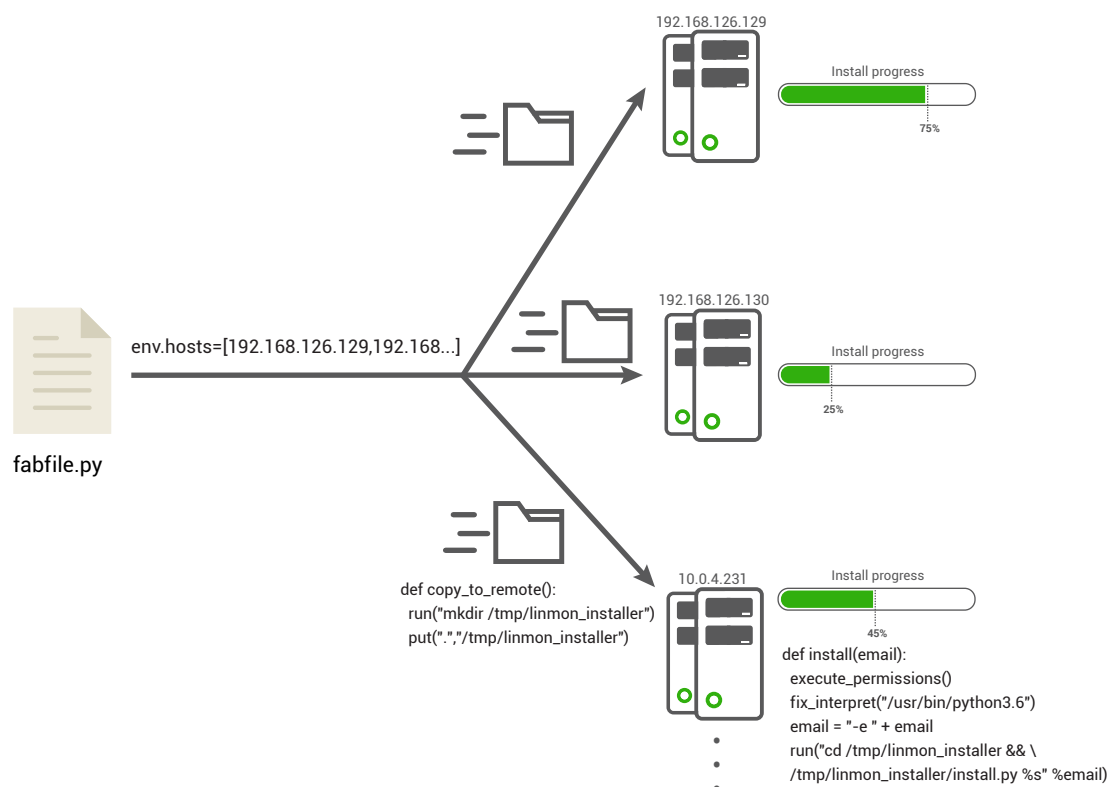
rôzne záznamy so vzormi, ktoré nemohli byť obsiahnuté v implementácii z dôvodu nedostatku relevantných dát syslog na skúmanie.

5.4 Podporné skripty

Skripty v zložke `aux_scripts` slúžia k minoritným nastaveniam. Skript `python_shebang.sh` slúži na hromadnú zmenu shebangu vo všetkých python skriptoch. Tento skript vznikol kvôli nejednotnosti symbolického odkazu na najnovší interpret jazyku Python 3, pretože v systémoch založených na RHEL meno symbolického odkazu pozostáva nielen z majoritného ale aj s minoritného čísla verzie oproti systémom založených na Debiane, ktoré majú jednoducho symbolický odkaz `python3`. Druhý skript je vytvorený na testovacie účely, pričom nastavuje konfiguračný súbor pre `sendmail`, tak aby bolo možné používať zabezpečenú komunikáciu cez SMTP relay.

5.5 Nasadenie pomocou Fabric3

Na nasadenie monitorovacieho nástroja pre testovacie účely bola zvolená knižnica Fabric3, ktorá je portom knižnice Python Fabric a zabezpečuje chod pod Python 3. Skript `fabfile.py` obsahuje funkcie potrebné na nakopírovanie jednotlivých zložiek do monitorovaných zariadení, nakonfigurovanie a inštaláciu nástroja Linmon. Na sledujúci obrázok 5.6 zobrazuje princíp nasadenia s úryvkami vlastného kódu.



Obr. 5.6: Schéma práce nasadzovacieho skriptu

Kapitola 6

Testovanie

Táto kapitola sa venuje testovaniu či už nástroju ako celku, tak aj testovaniu jednotlivých vybraných kľúčových častí hlavného skriptu. V neposlednom rade sa taktiež zaoberá testovaním filtrovania notifikačných správ, čo patrí k nosným bodom celej tejto práce. Výsledky testovania jednotlivých monitorovacích modulov nie sú súčasťou tejto kapitoly, nakoľko boli testované počas implementácie a vzhľadom k množstvu skriptov by rozsah tejto kapitoly značne znásobili. Po dôkladnom zvážení bolo vypustené testovanie užívateľského rozhrania nakoľko primárnymi používateľmi skriptov sú pokročilí užívatelia – systémoví administrátori, preto sa testovanie sústreďuje skôr na kľúčové prvky skriptov.

6.1 Testovacie prostredie

Testovanie všetkých súčastí programu, teda hlavného skriptu a monitorovacích modulov prebiehalo v dvoch typoch zariadení, a to na fyzickom a virtuálnom, jednotlivé testovacie prostredia môžeme vidieť v tabuľkách 6.1, 6.2, 6.3, 6.4. Ako si môžeme všimnúť, tak na jednotlivých testovacích strojoch boli nainštalované rôzne distribúcie GNU/Linux s rôznymi init systémami, a to umožnilo otestovanie funkčnosti na **SystemD**, **SysV** a **Upstart**. Navyiac bola otestovaná prenositeľnosť nástroju Linmon naprieč distribúciami založenými na RHEL a Debian. Pre skrátenie veľkosti tabuliek bola zo zoznamu odstránená absolútna cesta ku skriptom `/opt/linmon/monitoring_scripts`.

Na fyzickom zariadení z tabuľky 6.1 boli testované prevažne monitorovacie moduly monitorujúce disky. Tento prístup bol zvolený z dôvodu, že na virtuálnych zariadeniach nie je možné odčítať hodnoty S.M.A.R.T a taktiež teplotu virtuálnych diskov. Navyiac bola pri monitorovaní zmenená IP adresa rozhrania a pripojený disk, ktorý nebol špecifikovaný ako povolený, čo malo za následok notifikáciu administrátora.

CPU	Intel i5 6200 2 cores/ 4 threads
RAM	8192MB
HDD/SSD	500GB HDD NTFS, 256GB SSD (113GB EXT4)
OS	Debian 10 64bit, SystemD, Kernel 4.15.0
NET	lo (loopback), enp0s31f6 (ethernet), wlp3s0 (Wi-Fi)
Linmon CRON	5 ./user_cpu_usage.py mysql -t 10 -i error 5 [DISK] ./disk_temp.py -a -t 30 -i warning 5 [DISK] ./disk_allow_serial.py -i warning 5 [AUTH] ./rsyslog_auth.py -i critical 10 [NETCARD] ./netcard_ip_change.py -i error 15 [DISKUSE] ./disk_usage.py -b /dev/sda1 -p 10 -i critical 00:00 [SMART] ./disks_smart_mon.py -i critical 60 [DISKUSE] ./disk_usage.py -b /dev/sda1 -p 10 -i critical

Tabuľka 6.1: Fyzický stroj

Virtuálny stroj č.1 bol webový server, pri ktorom boli monitorované služby s nim súvisiace, a to démon webserveru Apache a databázy Mysql. Boli vyvolané podmienky na notifikáciu pri vypnutí jednej z týchto služieb, následne bola vyvolaná žiadosť o prístup k `/etc/passwd`, čo spôsobilo notifikáciu za pomoci skriptu `rsyslog_apache.py`. Nakoniec bolo vypnuté virtuálne zariadenie s IP adresou `192.168.126.131`, čo vyvolalo príslušnú notifikáciu.

CPU	2 cores
RAM	2048MB
HDD/SSD	20GB vmdk EXT4
OS	Centos 7 64bit, SystemD, Kernel 3.10.0
NET	lo (loopback), ens33 (ethernet)
Linmon CRON	5 ./user_cpu_usage.py mysql -t 10 -i error 5 [APACHE] ./proc_ram_mon.py apache -p 40 -i warning 5 [APACHE] ./proc_cpu_mon.py apache -p 50 -i info 5 [APACHE] ./rsyslog_apache.py -i error 2 [SERVICE] ./service_alive_stat.py -i critical -a httpd 2 [SERVICE] ./service_alive_stat.py -i critical -a mysqld 1 ./node_alive.py -s 192.168.126.131 -t 4 -i critical

Tabuľka 6.2: Virtuálny stroj 1

Na virtuálnom stroji č.2 bolo monitorované vyťaženie hardvéru, pričom simulácia tohto vyťaženia bola realizovaná programom **stress**. Napokon boli monitorované služby **postfix**, **named** a **sendmail** a notifikácia bola vyvolaná ich zastavením. Tak ako v predchádzajúcom virtuálnom stroji, tak aj v tomto sa simulovalo vypnutie susedného zariadenia s následnou notifikáciou o jeho neaktívnosti.

CPU	2 cores
RAM	4096MB
HDD/SSD	20GB vmdk EXT4
OS	Ubuntu 14.04 64bit, Upstart, Kernel 4.4.0
NET	lo (loopback), eth0 (ethernet)
Linmon CRON	5 [RES] ./ram_free_mon.py -p 20 -i error 5 [RES] ./cpu_mon.py -t 80 -i critical 5 [RES] ./swap_mon.py -p 20 -i critical 2 [SERVICE] ./service_alive_stat.py -i critical -a postfix 2 [SERVICE] ./service_alive_stat.py -i critical -a sendmail 2 [SERVICE] ./service_alive_stat.py -i critical -a named 1 ./node_alive.py -s 192.168.126.129 -t 4 -i warning

Tabuľka 6.3: Virtuálny stroj 2

Na poslednom testovacom stroji č.3 bol podobne ako na predchádzajúcom monitorovaný hardvér a živosť ďalšieho susedného stroja. Navyše tento virtuálny stroj obsahoval softvérové RAID polia typu 0, 1 a 5, na ktorých boli zámerne vyvolané chyby s cieľom otestovať notifikáciu pomocou emailu. Poslednou testovanou časťou bola notifikácia nových pripojených USB zariadení, ktoré by mohli byť v reálnej prevádzke nechcené a obsahovať škodlivé kódy.

CPU	2 cores
RAM	4096MB
HDD/SSD	20GB vmdk EXT4
OS	Devuan Jessie 1.0 64bit, SysV, Kernel 3.16.0
NET	lo (loopback), eth0 (ethernet)
Linmon CRON	5 [RES] ./ram_free_mon.py -p 20 -i error 5 [RES] ./cpu_mon.py -t 80 -i critical 5 [RES] ./swap_mon.py -p 20 -i critical 2 [SERVICE] ./service_alive_stat.py -i critical -a cupsd 1 ./node_alive.py -s 192.168.126.130 -t 4 -i warning 10 [NEWDEVS] ./dev_usb_new.py -i info 10 [NEWDEVS] ./disk_allow_uuid.py -i warning 60 ./disk_health_mdadm -a -i error

Tabuľka 6.4: Virtuálny stroj 3

6.2 Testovanie mechanizmu vysokej dostupnosti

Ako už bolo spomenuté v predchádzajúcich kapitolách, tak na zaistenie stáleho chodu programu sú použité dva od seba nezávislé prístupy. Prvým testovaným prístupom je reštartovanie služby pri výpadku za pomoci init systému. Na nižšie uvedenom výpise procesov môžeme vidieť bežiaci proces s PID 372.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	372	1	0	08:16	?	00:00:00	/usr/bin/python3 /opt/linmon/linmon.py

Na simuláciu výpadku bol zadaný príkaz `kill -9 372`, ktorý spôsobil zastavenie bežiacieho procesu. Bezodkladne po zastavení procesu sa monitorovací nástroj znovu spustí, čo dokazuje aj nasledovný výpis z bežiacich procesov.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	398	1	0	08:16	?	00:00:00	/usr/bin/python3 /opt/linmon/linmon.py

Druhý spôsob zaistujúci nepretržitý beh, záznam v plánovači `cron`, bude spustený len v prípade zlyhania skriptu init systému, prípadne pokiaľ by predbehol proces reštartu init systémom.

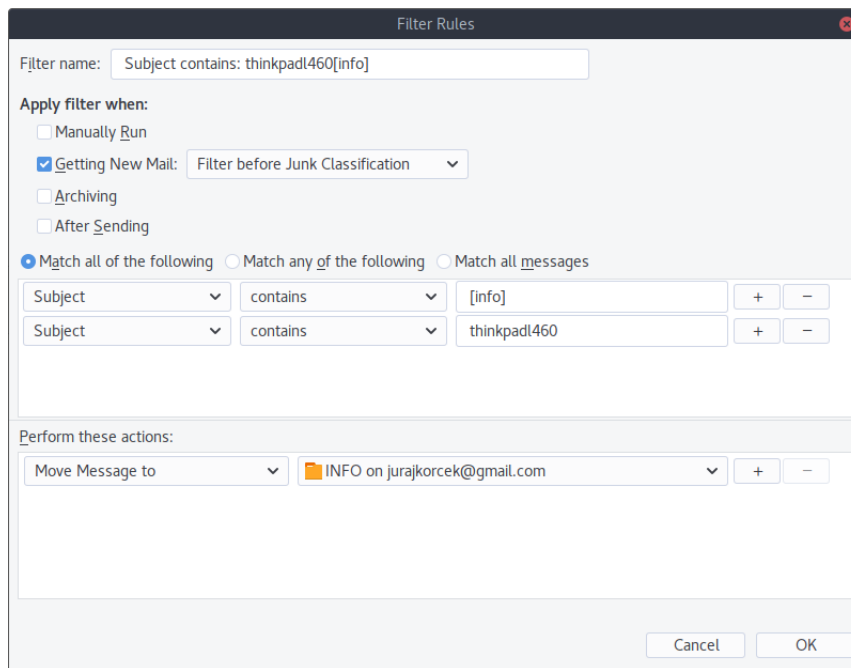
UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	639	1	0	08:29	?	00:00:00	/usr/bin/python3 /opt/linmon/linmon.py

Po spustení monitorovania má skript PID 639 a po aplikovaní príkazu `kill -9 639` sa proces monitorovania skriptu zastaví. Následne záznam v crone zabezpečí opätovné spustenie, čo môžeme vidieť na výpise nižšie.

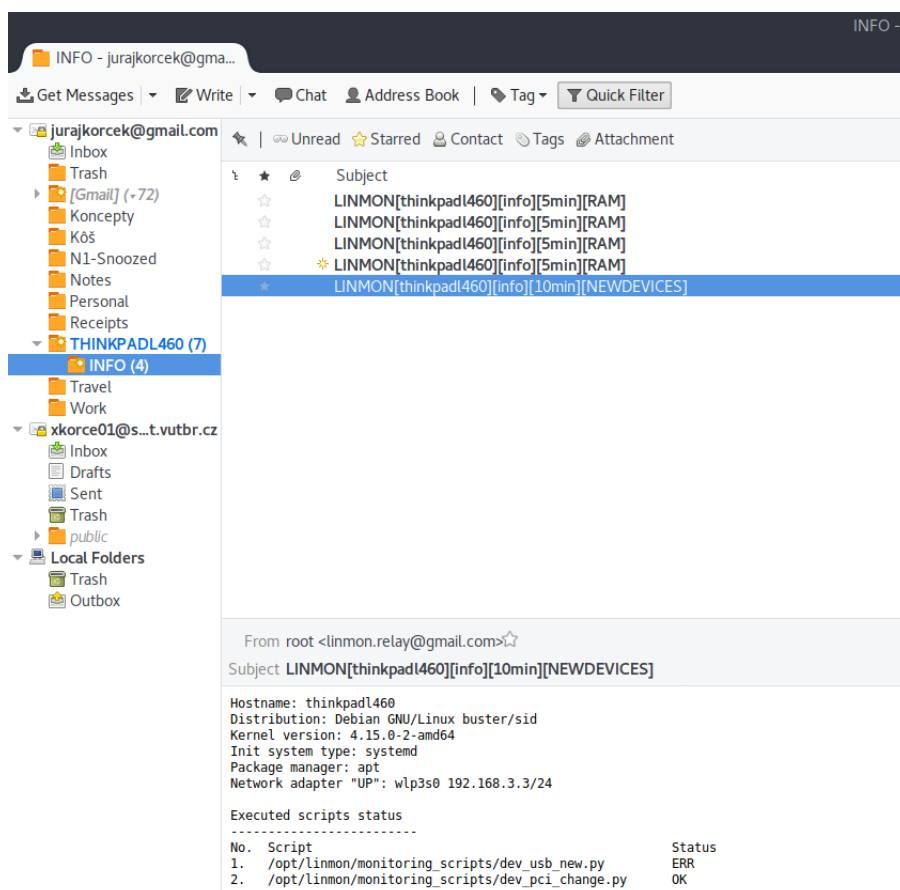
UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	681	679	0	08:30	?	00:00:00	/bin/sh -c /opt/linmon/linmon.py
root	682	681	4	08:30	?	00:00:00	/usr/bin/python3 /opt/linmon/linmon.py

6.3 Filtrovanie notifikačných správ

Testovanie filtrovania emailových správ prebiehalo na zariadení z tabuľky 6.1 pričom pre prijímanie správ bol použitý program Mozilla Thunderbird v. 52.5.2. Na obrázku 6.1 je definované pravidlo pre filtrovanie správ, v ktorých predmet obsahuje najnižšiu možnú úroveň závažnosti notifikácie, pričom pri prijatí správy so značkou **info** premiestni túto správu do separátneho priečinku. Priečinkov s vyfiltrovanými správami, ktoré obsahujú značku **info** môžeme vidieť na obrázku 6.2. Keďže predmet správy obsahuje ďalšie dôležité značky ako napríklad `hostname`, skupinu z plánovača a oneskorenie spustenia skriptov, prípadne čas spustenia, tak je možné na základe všetkých týchto značiek logicky roztriediť notifikačné správy. Navyše je možné správy vyfiltrovať aj podľa parametrov monitorovaného zariadenia z tela správy. Notifikačné správy je teda možné bez problémov organizovať či už podľa závažnosti správ, monitorovaných súčastí systému, frekvencie spúšťania skriptov a rôznych iných vlastností.



Obr. 6.1: Nastavenie filtrovania prijatých správ



Obr. 6.2: Vyfiltrované správy v separátnom priečinku

6.4 Testovanie módu údržby

Mód údržby slúži na pozastavenie monitorovania, preto je nutné otestovať jeho funkčnosť, a to predovšetkým notifikáciu o začiatku a konci údržby a pozastavenie zasielania správ. Na nasledujúcom výpise z prijatých správ môžeme vidieť, že o 9:01 bol aktivovaný mód údržby pomocou príkazu `./linmon.py -m on`.

Subject	Correspondent	Date
.		
.		
.		
LINMON[thinkpadl460] [warning] [5min] [DEFAULT]	root	26.4.2018 8:41
LINMON[thinkpadl460] [info] [5min] [RAM]	root	26.4.2018 8:41
LINMON[thinkpadl460] [info] [5min] [RAM]	root	26.4.2018 8:46
LINMON[thinkpadl460] [info] [10min] [NEWDEVICE]	root	26.4.2018 8:46
LINMON[thinkpadl460] [info] [5min] [RAM]	root	26.4.2018 8:51
LINMON[thinkpadl460] [error] [15min] [DISKUSE]	root	26.4.2018 8:51
LINMON[thinkpadl460] Maintenance mode ON	root	26.4.2018 9:01

Mód údržby bol spustený po dobu 50 minút, kedy sa zadal príkaz na jeho vypnutie `./linmon.py -m off`, následne, ako môžeme vidieť na výpise nižšie, tak monitorovanie plynulo pokračovalo ďalej.

Subject	Correspondent	Date
.		
.		
.		
LINMON[thinkpadl460] [error] [15min] [DISKUSE]	root	26.4.2018 8:51
LINMON[thinkpadl460] Maintenance mode ON	root	26.4.2018 9:01
LINMON[thinkpadl460] Maintenance mode OFF	root	26.4.2018 9:51
LINMON[thinkpadl460] [warning] [5min] [DEFAULT]	root	26.4.2018 9:56
LINMON[thinkpadl460] [info] [5min] [RAM]	root	26.4.2018 9:56
LINMON[thinkpadl460] [info] [5min] [RAM]	root	26.4.2018 10:01

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bol návrh a následná implementácia monitorovacích skriptov, ktoré budú sledovať stav počítačových clusterov s následnou notifikáciou systémových administrátorov. Implementačným jazykom bol skriptovací jazyk Python 3 pričom pri implementácii bol braný ohľad na minimálny počet programov, ktoré je nutné nainštalovať na zaistenie práce monitorovacieho nástroja. Z tohoto dôvodu sú ako notifikačný prostriedok použité emailové správy, a teda jedinými dvoma nutnými programami pre chod hlavného monitorovacieho a notifikačného skriptu sú interpret jazyka Python 3.5 alebo vyšší a program `sendmail`.

Výsledný monitorovací program vylepšuje metódu monitorovania tým, že oproti štandardnému prístupu, kde sú v plánovači `cron` zadané skripty na monitorovanie a každý skript posiela emailovú správu, umožňuje program `Linmon` zoskupovať výstupy viacerých skriptov do jednej správy a tým nezahľcovať emailovú schránku. Navyše umožňuje efektívne filtrovanie správ na základe značiek v predmete správy. Oproti systémom akými sú Nagios a Zabbix nie je nutná inštalácia balíčkov pre hlavný monitorovací skript, nakoľko inštalácie v niektorých proprietárnych systémoch môžu byť zakázané. Okrem vyššie spomenutých vlastností bol skript vyvíjaný s ohľadom na prenositeľnosť naprieč rôznymi distribúciami, a to podporou init systémov `SystemD`, `SysV` a `Upstart` ako aj podporou rôznej hierarchie súborového systému. Taktiež umožňuje jednoduchú modifikáciu už existujúcich skriptov a tvorbu nových, čomu dopomáha ako návod, tak aj vzorové skripty.

Monitorovacie moduly umožňujú zisťovanie vysokých záťaží zdrojov zariadenia, monitorovanie záznamov `syslog` a monitorovanie služieb systému, ktoré poskytnú výstup hlavnému skriptu, ktorý môže zoskupiť viacero výstupov do jednej správy a následne notifikuje administrátora.

Kvôli implementácii bolo nutné naštudovať záznamy `syslog` rôznych známych a často používaných programov v serverových systémoch GNU/Linux. Táto časť riešenia bakalárskej práce bola jednou z najobtiažnejších, čo sa týka získavania informácií, nakoľko výstupy záznamov `syslog` nie sú jednoducho získateľné kvôli citlivým dátam. Z tohto dôvodu bol počet skriptov monitorujúcich záznamy `syslog` o niečo zmenšený oproti pôvodnému návrhu.

Do budúcnosti sa počíta s rozšírením počtu skriptov monitorujúcich záznamy `syslog` a taktiež optimalizácií filtrovacích značiek umožňujúcich triedenie notifikačných emailov na základe reálneho používania. Ďalším možným rozšírením implementovaným v budúcnosti bude pridanie podpory pre `journalctl`, ktorý sa používa v distribúcii Fedora na prehliadanie záznamov `syslog`.

Literatúra

- [1] *The Syslog Protocol*. Marec 2009, [Online; navštívené 05.01.2018].
URL <https://tools.ietf.org/html/rfc5424>
- [2] Allen, M.: *What a Linux Distribution Is*. Jún 2000, [Online; navštívené 15.01.2018].
URL <http://www.comptechdoc.org/os/linux/articles/ardistro.html>
- [3] Bookman, C.: *Linux Clustering: Building and Maintaining Linux Clusters*. Sams, 2002, ISBN 1578702747.
- [4] Forcier, J.: *Fabric - Pythonic Remote Execution*. [Online; navštívené 10.12.2017].
URL <http://docs.fabfile.org/>
- [5] Foundation, T. L.: *Filesystem Hierarchy Standard*. Marec 2015, [Online; navštívené 15.04.2018].
URL https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html
- [6] H., S. C.: *A Byte of Python*. CreateSpace Independent Publishing Platform, 2015, ISBN 978-1-5148-2814-4.
- [7] Hoch, D.: *Linux System and Performance Monitoring*. Darren Hoch, 2009.
- [8] Johnson, S. K.: *Performance Tuning for Linux Servers*. IBM Press, 2005, ISBN 0137136285.
- [9] Lee, J.: *LINUX 3 UNIX-Like Operating Systems That Aren't Linux*. August 2015, [Online; navštívené 10.01.2018].
URL <https://www.makeuseof.com/tag/3-unix-like-operating-systems-arent-linux/>
- [10] Pillai, S.: *What is Nagios: An Introduction to enterprise level server monitoring*. Jún 2013, [Online; navštívené 12.12.2017].
URL <https://www.slashroot.in/what-nagios-introduction-enterprise-level-server-monitoring>
- [11] Prakash, A.: *Linux is Running on Almost All of the Top 500 Supercomputers*. September 2017, [Online; navštívené 16.01.2018].
URL <https://itsfoss.com/linux-supercomputers-2017/>
- [12] Red Hat, Inc.: *Ansible Documentation*. [Online; navštívené 10.12.2017].
URL <http://docs.ansible.com/>
- [13] Selenic: *smem memory reporting tool*. [Online; navštívené 25.02.2018].
URL <https://www.selenic.com/smem/>

- [14] Silberschatz, A.; Galvin, P.; Gagne, G.: *Operating System Concepts*. John Wiley & Sons, 2008, ISBN 978-1-118-06333-0.
- [15] Stallman, R.: *Linux a systém GNU*. [Online; navštívené 19.12.2017].
URL <https://www.gnu.org/gnu/linux-and-gnu.html>
- [16] Turnbull, J.; Matotek, D.; Lieverdink, P.: *Pro Linux System Administration*. Apress, 2009, ISBN 978-1-4302-1913-2.
- [17] Vaghasia, R.: *Top 5 High Availability Dedicated Server Solutions*. Jún 2017, [Online; navštívené 15.01.2018].
URL <https://www.accuwebhosting.com/blog/top-5-high-availability-dedicated-server-solutions/>
- [18] Venus, C.: *Free, secure, easy - Linux as an alternative to Windows and Mac*. Júl 2016, [Online; navštívené 19.01.2018].
URL <http://technology.inquirer.net/50254/linux-operating-system-personal-computer-software>
- [19] Williams, E.: *What is the Meaning of Deployment in Software*. December 2017, [Online; navštívené 12.01.2018].
URL <https://pdf.wondershare.com/business/what-is-software-deployment.html>
- [20] Zabbix LLC: *Zabbix Documentation*. [Online; navštívené 13.12.2017].
URL <http://www.zabbix.com/documentation>
- [21] Žák, K.: *Historie OS UNIX*. Jún 2001, [Online; navštívené 11.01.2018].
URL <https://www.root.cz/clanky/historie-os-unix/>

Príloha A

Zoznam a činnosť implementovaných skriptov

Názov skriptu	Účel	Výstup
dev_pci_change	odpojené alebo pripojené zariadenia pci	notifikuje administrátora o zmenách zariadení
dev_usb_new	nové pripojené zariadenia usb	notifikuje administrátora o zmenách zariadení
disk_allow_serial	špecifikuje povolené disky na základe sériového čísla	notifikuje pri pripojení iného zariadenia, ako je povolené
disk_allow_uuid	špecifikuje povolené disky na základe UUID	notifikuje pri pripojení iného zariadenia, ako je povolené
disk_health_hp	monitoruje stav diskových polí HP, fyzické a logické disky	pri abnormálnej hodnote informuje administrátora
disk_health_mdadm	monitoruje stav diskových polí vytvorených pomocou MDADM	pri abnormálnej hodnote informuje administrátora
disks_smart_mon	monitoruje hodnoty S.M.A.R.T	pri nájdení abnormálnej hodnoty informuje administrátora
disk_temp	monitoruje teplotu diskov	pri zvýšených hodnotách informuje administrátora o konkrétnom zariadení a teplote
disk_temp_hp	monitoruje teplotu diskových polí HP	pri zvýšených hodnotách informuje administrátora o konkrétnom zariadení a teplote
disk_usage	sleduje zaplnenie diskov	po prekročení zadaného prahu notifikuje administrátora
service_alive_stat	kontrola aktivity/pasivity definovaných služieb	pokus o spustenie v prípade (ne)aktívnosti, notifikácia administrátora
node_alive	kontrola chodu/aktivity ďalších počítačov/clusterov	v prípade neaktívnosti notifikácia administrátora

Názov skriptu	Účel	Výstup
netcard_ip_change	monitoruje zmenu IP adresy	pri zmene IP adresy notifikuje administrátora
netcard_ip6_change	monitoruje zmenu IPv6 adresy	pri zmene IPv6 adresy notifikuje administrátora
netcard_mac_change	monitoruje zmenu MAC adresy	pri zmene MAC adresy notifikuje administrátora
user_cpu_usage	monitoruje zaťaženie procesora užívateľom	pri prekročení prahu informuje administrátora
user_ram_usage	monitoruje vyťaženie RAM užívateľom	pri prekročení prahu informuje administrátora
group_cpu_usage	monitoruje zaťaženie procesora používateľov z definovanej skupiny	pri prekročení prahu informuje administrátora
group_ram_usage	monitoruje vyťaženie RAM používateľov z definovanej skupiny	pri prekročení prahu informuje administrátora
proc_cpu_mon	monitoruje zaťaženie procesora definovaným procesom	pri prekročení prahu informuje administrátora
proc_ram_mon	monitoruje využitie RAM definovaným procesom	pri prekročení prahu informuje administrátora
cpu_mon	monitoruje zaťaženie procesora	pri prekročení prahu notifikuje administrátora
ram_free_mon	monitoruje množstvo voľnej pamäti RAM	pri poklese pamäti pod definovaný prah notifikuje administrátora
ram_used_mon	monitoruje množstvo zabranej pamäti RAM	pri prekročení definovaného prahu notifikuje administrátora
swap_mon	monitoruje množstvo voľného miesta v SWAP	pri poklese pamäti pod definovaný prah notifikuje administrátora
rsyslog_apache	“Client sent malformed Host header”	notifikácia administrátora
	Resource temporarily unavailable: fork	
	Pokus o prístup k /etc/passwd	
rsyslog_auth	3 incorrect password attempts	notifikácia administrátora
	Failed password for	
	FAILED su for	
	Illegal user	
	Invalid user	
rsyslog_smart	Did not receive identification string from	notifikácia administrátora
	Pending Sectors	
	Uncorrectable errors	
rsyslog_smart	Prefailure attribute	notifikácia administrátora
	Username/Password verification failed	

Názov skriptu	Účel	Výstup
rsyslog_dovecot	imap-login: Aborted login	notifikácia administrátora
	pop3-login: Aborted login	
	Can't contact LDAP server	
	Can't connect to server	
rsyslog_postfix	Improper use of SMTP command pipelining	notifikácia administrátora
	Insufficient system storage	
rsyslog_postgre	password authentication failed for user	notifikácia administrátora
rsyslog_sendmail	rejected commands from x.x.x.x due to pre-greeting traffic	notifikácia administrátora
	Connection rate limit exceeded	
	rejecting connections on daemon MTA: load average	
	runqueue: Aborting queue run: load average too high	
rsyslog_universal	Hľadá zhodu s definovaným regulárnym výrazom v špecifikovanom súbore syslog zadanými pomocou prepínačov. Slúži na nájdenie akéhokoľvek vzoru z ľubovoľného syslog súboru, ktorý nie je definovaný skriptami popísanými vyššie.	pri nájdení zhody notifikácia administrátora

Príloha B

Obsah CD

- `src/`
 - `aux_scripts/` – Pomocné skripty.
 - `configs/` – Konfiguračné súbory pre skripty.
 - `example_scripts/` – Skripty s komentármi na demonštráciu vytvorenia nových skriptov.
 - `init_scripts/` – Skripty pre init systémy.
 - `monitoring_scripts/` – Monitorovacie moduly.
 - `packages/` – Podporné moduly.
 - `syslog_scripts/` – Moduly monitorujúce záznamy syslog.
 - `fabfile.py` – Python Fabric skript na nasadenie.
 - `install.py` – Skript zabezpečujúci inštaláciu a odinštaláciu.
 - `linmon.py` – Hlavný monitorovací a notifikačný skript.
 - `README.md` – Návod na inštaláciu, prácu s nástrojom a tvorbu nových skriptov.
- `doc/`
 - `src/` – Zdrojové súbory technickej správy.
 - `xkorce01-Monitorovani-behu-OS-Linux.pdf` – Technická správa vo formáte PDF.